Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Matematica

CURVE E SUPERFICI DI SUDDIVISIONE CON APPLICAZIONE ALLE IMMAGINI DIGITALI

Tesi di Laurea in Grafica

Relatore:
Chiar.mo Prof.
Giulio Casciola

Presentata da:
Lucia Capecci

I Sessione
Anno Accademico 2011/2012

A Dani e Cate

oltre ai miei fantastici genitori
a Matteo
e a tutta la mia grande Famiglia

Introduzione

Le curve e le superfici di suddivisione sono un importante strumento a disposizione della Computer Graphics. Sono dei metodi di raffinamento facili da implementare, computazionalmente efficienti e permettono la definizione di curve e superfici con un certo grado di regolarità. In questa tesi si vuole approfondire il concetto di schema di suddivisione e si vogliono classificare gli schemi per curve. Si vogliono, poi, definire degli schemi di suddivisione bivariati e dare l'idea di come tali schemi possono essere utilizzati in particolari applicazioni per immagini digitali.
Scopo della tesi è quello di esplorare nuove tipoogie di applicazioni per gli schemi di suddivisione, in particolare si sono indivisuati i problemi di scaling e compressione di immagini digitali affrontati in letteratura con metodi completamente differenti.
La tesi è divisa in quattro capitoli. Nel primo capitolo, dopo averne definito tutte le proprietà, si classificano gli schemi di suddivisione per curve. Nel secondo capitolo si spiegano gli schemi di suddivione bivariati e si presentano i problemi dello scaling e della compressione di segnali e di immagini. Si propongono, quindi, degli schemi univariati per la compressione di segnali e schemi bivariati per lo scaling e compressione di immagini.
Durante il lavoro di tesi si sono realizzati diversi codici di calcolo che permettono di utilizzare gli schemi di suddivisione per curve, di effettuare uno zoom-in di un'immagine digitale, di comprimere sia un segnale che un'immagine. Nel terzo capitolo si descrive l'implementazione in ambiente Matlab dei vari schemi di suddivisione visti e delle loro applicazioni e sono presenti delle piccole guide su come utilizzare le intefacce grafiche create a supporto dei codici.
Infine, l'ultimo capitolo è dedicato alla sperimentazione. Sono quindi presenti, oltre ai risultati e agli esempi degli schemi di suddivisione per curve, anche i risultati dello scaling, della compressione di segnali e immagini.

Indice

Capitolo 1

Classificazione schemi di suddivisione

Gli schemi di suddivisione sono degli efficienti metodi computazionali utilizzati per il disegno e la rappresentazione di curve e superfici. In questo capitolo si parlerà, in particolare, di schemi di suddivisione per curve, ottenuti semplicemente da rappresentazione vettoriale di *schemi univariati*. Il motivo principale per cui si studiano i metodi di suddivisione per curve è lo sviluppo di nuove tecnologie, facili da utilizzare ed implementare, che permettono la costruzione di curve più efficiente e più flessibile.
Il procedimento generale per la costruzione di una curva è il seguente: il designer definisce, in $\mathbb{R}^2$, un set di punti, chiamati *punti di controllo o punti iniziali*. Dove tali punti sono rappresentati come una sequenza di indici $(P^0, P^1, .., P^N) \in \mathbb{R}^2$, cioè i punti definiti sono in $\mathbb{R}^2$ ma la sequenza che li rappresenta è un vettore. Si definisce la *poligonale di controllo o iniziale* come la curva lineare a tratti passante attraverso i punti di controllo. L'obiettivo dello schema di suddivisione è quello di rappresentare
matematicamente una curva in $\mathbb{R}^2$ che segua in forma la poligonale di controllo o interpolando i punti di controllo o, semplicemente, approssimandone la forma. I metodi di suddivisione forniscono un raffinamento della poligonale. Dopo un certo numero di successivi raffinamenti, il metodo di suddivisione definisce una curva, chiamata *curva limite*, con un certo grado di regolarità.
Diversi ricercatori hanno concentrato i loro studi sui metodi di suddivisione e, nelle seguenti tabelle sono riportati i lavori più importanti.

Autori	**Articolo**	**Anno**
de Rham, G.	Trisection Algorithms of 1947,'53,'56,'57,'58	1947
Chaikin, G. M.	An algorithm for high speed curve generation	1974
Riesenfeld, R.	On Chaikinis Algorithm	1975
Catmull & Clark	Recursively generated B-Spline Surfaces on arbitrary topological meshes	1978
Doo & Sabin	Behavior of recursive division surfaces near extraordinary points	1978
Lane & Riesenfeld	Uniform B-splines of arbitrary order	1980
de Boor	Corner-Cutting always works	1987
Gregory & Qu	Non-uniform corner-cutting	1988
de Boor	Local corner cutting and smoothness of limit curve	1990
Dyn, Gregory & Levin	Analysis of uniform binary subdivision	1991
Dubuc	Interpolation through an iterative scheme	1986
Dyn, Levin & Gregory	A 4-point interpolatory subdivision scheme	1987
Gregory & Qu	Non-uniform corner-cutting	1988
Deslauries & Dubuc	Symmetric Iterative Refinement	1989
Dyn, Gregory & Levin	Buttefly subdivision for surfaces	1990
Kobbelt	A variational approach to subdivision	1996
Kobbelt	Discrete Fairing (for surfaces)	1997
Kuijt & van Damme	Convexity preserving interpolatory subdivision	1998
Levin	Analysis of non-uniform binary subdivision	1999
Marinov, Dyn & Levin	Geometrically controlled 4-point schemes	2004

Dalla tabella si nota che lo studio degli schemi di suddivisione è molto recente e dai vari articoli si ha che esistono diverse tipologie di schemi di suddivisione, ad esempio alcuni schemi interpolano la poligonale di controllo, alcuni che la approssimano, degli schemi particolari hanno la proprietà di riprodurre delle coniche, oppure le curve limite sono di una certa regolarità. L'obiettivo di questo capitolo è di presentare gli schemi di suddivisione e di classificarli a seconda delle proprietà. Inoltre, sono presenti diversi esempi di schemi che possono essere confrontati tra loro.

Uno *schema di suddivisione* è un processo che definisce delle curve regolari come limite di successivi raffinamenti. Partendo da una sequenza iniziale di punti in $\mathbb{R}^2$, detta *poligonale di controllo*, si definiscono nuovi punti ad ogni passo di raffinamento, in modo che dopo diversi passi si avrà una curva limite regolare.

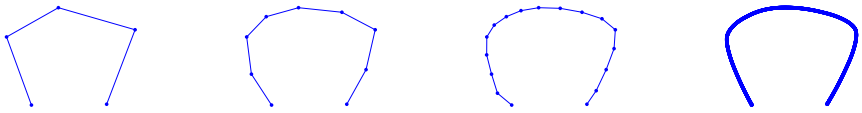


Figura 1.1: Poligonale iniziale; primo raffinamento; secondo raffinamento e curva limite

Si parla di schemi *binary* e *ternary*: nel primo tipo ad ogni passo si aggiunge un solo punto tra due punti del livello precedente, nel secondo caso se ne aggiungono due.

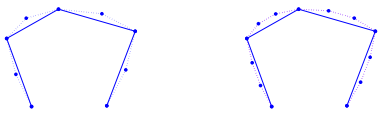


Figura 1.2: Schema binary e schema ternary

Si può pensare di aggiungere più punti ma si aggira questo problema combinando opportunamente i due tipi di schemi, ad esempio per aggiungere quattro punti basta applicare uno schema binary due volte e così via.
Gli schemi considerati in questa classificazione sono binary, ma tutte le proprietà possono essere estese anche alla teoria degli schemi ternary. Per questo motivo se i punti iniziali sono N al primo raffinamento, diventeranno $2N-1$, al secondo $4N-3$ e al k-esimo $2kN-2k-1$.
Dato P^0 un vettore di dimensione N di punti inziali P_i^0 $i=1,...,N$ si ha che la *legge di raffinamento relativa al primo passo* dello schema di suddivisione è definita come una matrice $\mathbf{S}_0$ di dimensione $(2N-1)\times N$ che permetta di costruire il nuovo vettore raffinato $P^1=\{P_i^1\}_{i=1,...,2N-1}$ di dimensione $2N-1$

$$P^1=\mathbf{S}_0P^0$$

Si hanno, quindi, le leggi di raffinamento relative a tutti i livelli in modo che

$$P^{k+1}=\mathbf{S}_kP^k$$

in cui le dimensioni sono: $2kN-2k-1$ per il vettore P^k, $(2kN-2k-1)\times(2(k+1)N-2(k+1)-1)$ per la matrice $\mathbf{S}_k$ e $2(k+1)N-2(k+1)-1$ per il vettore P^{k+1}.
Esistono due categorie principali in cui classificare gli schemi di suddivisione: gli schemi di approssimazione e gli schemi di interpolazione.

Definizione 1.1. Uno schema si dice di *approssimazione* se approssima la forma della poligonale di controllo, cioè se approssima i punti di controllo senza interpolarli.
Uno schema si dice di *interpolazione* se i punti del livello k-esimo appartengono anche al vettore del livello $(k+1)$-esimo, cioè se la curva limite passa per i punti del vettore iniziale.

Figura 1.3: Approssimazione e Interpolazione

Negli schemi interpolanti si possono considerare i nuovi punti pari e dispari separatamente: per i nuovi punti pari ci sarà bisogno della legge di raffinamento mentre, per quelli dispari, basta fare una replica dei punti al livello precedente. La matrice $\mathbf{S}_k$, cioè, avrà le righe pari con i coefficienti che definiscono la legge e le righe dispari saranno rappresentate dalla base canonica dello spazio di riferimento. In pratica:

$$\mathbf{S}_k = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ s_{2,1} & s_{2,2} & \dots & & \dots \\ 0 & 1 & 0 & 0 & \dots \\ s_{4,1} & s_{4,2} & \dots & & \dots \\ 0 & 0 & 1 & 0 & \dots \\ & \dots & \dots & & \ddots \end{pmatrix}$$

Si osserva che il risultato del raffinamento è un vettore dato dal prodotto di una matrice per un vettore. Quindi ogni singola riga di posizione i della matrice $\mathbf{S}_k$ definisce il valore del punto i-esimo del vettore risultato. Cioè questa riga ha i coefficienti della combinazione che, applicata al vettore P^k definirà il punto P_i^{k+1}.

Definizione 1.2. Se uno schema di suddivisione prevede di combinare tutti i punti del vettore P^k nello stesso modo allora ogni riga di $\mathbf{S}_k$ deve avere gli stessi coefficienti, traslati opportunamente.
Uno schema si dice *stazionario* se la matrice $\mathbf{S}_k$ ha questa particolarità. La matrice non è mai la stessa perché ad ogni passaggio aumenta le dimensioni ma i suoi coefficienti sono solo traslati e non cambiano.
Invece, uno schema si dice *non stazionario* se la matrice $\mathbf{S}_k$ di raffinamento dipende dal passo k. Ad ogni passo i suoi coefficienti subiscono delle variazioni.

Definizione 1.3. Se su ogni riga della matrice $\mathbf{S}_k$ ci sono al più n elementi non nulli, lo schema determinerà il nuovo punto P_i^{k+1} con una combinazione di al più n punti P_j^k, in questo caso lo schema si dice *a n punti*.

Si definisce *partizione nodale* T una suddivisione dell'intervallo $[0, 1]$ in N intervalli tale che: $T = \{t_i \in [0, 1] : 0 = t_1 < t_2 < \cdots < t_N = 1\}$, dove ogni t_i è detto *nodo*. Una partizione nodale si dice *uniforme* se i nodi t_i sono equispaziati, viceversa, si dice *non uniforme*.
Si ha che ad ogni poligonale P^k è associata una partizione nodale in modo che i punti P_i^0 possano essere rappresentati parametricamente come $\{(t_i, P_i^k) : i \in \mathbb{Z}\}$. Con queste piccole premesse si può definire uno schema di suddivisione uniforme.

Definizione 1.4. Uno schema si dice *uniforme* se si possono rappresentare i punti come $\{(i2^{-k}, P_i^k) : i \in \mathbb{Z}\}$, cioè se i nodi associati ai punti sono equspaziati ad ogni passo di raffinamento.
Uno schema di suddivisione si dice *non uniforme* se i nodi della partizione associata non sono equispaziati. In questo caso i punti del raffinamento k-esimo hanno rappresentazione parametrica $\{(t_i, P_i^k) : i \in \mathbb{Z}\}$

Si ha che ogni volta che si aggiunge un nuovo punto si definisce anche un nuovo nodo. Nel caso uniforme il nuovo nodo è esattamente a metà dei due nodi relativi ai punti del passo precedente; nel caso non uniforme, invece, si distinguono due casi. Nel caso *semi-regolare* la partizione relativa alla poligonale iniziale non è uniforme ma si aggiungono i nuovi nodi nel punto medio dei nodi del passo precedente; nel caso *non regolare*, invece, il nuovo nodo è definito in una posizione arbitraria tra i nodi del passo precedente.

Definizione 1.5. Uno schema si dice *lineare* se ad ogni passo i nuovi punti sono definiti come una combinazione lineare dei punti al passo precedente. Se, invece, esiste un livello k in cui non esiste una definizione di questo tipo lo schema si dice *non lineare*.

Uno schema, banalmente, non può essere lineare e non lineare allo stesso tempo ma rientra in tutte le categorie precedenti, cioè è lineare o non lineare, stazionario o non stazionario etc. Per cui gli schemi si classificano in 16 tipologie diverse, date da tutte le possibili combinazioni delle precedenti proprietà. Di seguito si classificheranno tutti i

tipi di curve di suddivisione lineari e, come ultima categoria, si parlerà di due particolari esempi di schemi non lineari.

La maggioranza dei metodi di suddivisione può essere rappresentata dal seguente schema:

$$\begin{cases} P_{2i}^{k+1} = \sum_{j=0}^{m} s_{2i,j} P_{i+j}^{k} \\ P_{2i+1}^{k+1} = \sum_{j=0}^{m} s_{2i+1,j} P_{i+j}^{k} \end{cases} \quad k = 0, 1, 2... \tag{1.1}$$

con $s_{i,j}$ i coefficienti della matrice $\mathbf{S}$; con $s_{2i,j}$ relativi alle righe pari e $s_{2i+1,j}$ alle righe dispari.
Prima della trattazione è utile introdurre il concetto di *maschera k-esima* dello schema di suddivisione. Si definisce come l'insieme dei coefficienti di $\mathbf{S}$ relativi al passo k

$$a^k := \{s_{2i+1,0}, s_{2i,0}, s_{2i+1,1}, s_{2i,1}, \cdots, s_{2i+1,j}, s_{2i,j}, \cdots\} = \{a_0^k, a_1^k, \cdots\}$$

A ciascuna maschera è associato un *simbolo* definito come

$$a^k(z) := \sum_i a_i^k z^i \text{ con } z \in \mathbb{R}$$

Con queste informazioni preliminari si hanno gli strumenti per procedere con la classificazione degli schemi. Nelle seguenti pagine sono riportate le varie tipologie di metodi; la classificazione è divisa per sottosezione in cui si spiegano le caratteristiche principali degli schemi in questione e si forniscono degli esempi.

1.1. **Approssimazione Uniforme Stazionario**

Un generico schema di approssimazione uniforme e stazionario ha come legge:

$$\begin{cases} P_{2i}^{k+1} = \sum_{j=0}^{m} p_j P_{i+j}^{k} \\ P_{2i+1}^{k+1} = \sum_{j=0}^{m} d_j P_{i+j}^{k} \end{cases} \quad k = 0, 1, 2... \tag{1.2}$$

con p_j i coefficienti delle righe pari della matrice $\mathbf{S}$ e d_j i valori delle righe dispari.
I punti P_i^{k+1} sono una combinazione lineare dei punti al passo k-esimo quindi lo schema è lineare.
La maschera relativa a questo tipo di schemi é:

$$a := (d_0, p_0, d_1, p_1, \cdots, d_j, p_j, \cdots) = (a_0, a_1, \cdots)$$

Se lo schema considerato è convergente allora valgono le seguenti uguaglianze, [5]:

$$\begin{cases} \sum_j p_j = 1 \\ \sum_j d_j = 1 \end{cases} \quad \text{e} \quad \begin{cases} a(1) = 2 \\ a(-1) = 0 \end{cases} \tag{1.3}$$

Per questo tipo di schemi si prendono in considerazione le funzioni B-spline e si ha che il simbolo associato a tali funzioni di grado m è:

$$a(z) = 2^{-m}(1+z)^{m+1} \tag{1.4}$$

la legge di raffinamento si può scrivere come:

$$P_i^{k+1} = \sum_j a_{i-2j}^{[m]} P_j^k,\ i \in \mathbb{Z}; k = 0, 1, 2, ... \tag{1.5}$$

con

$$a_i^{[m]} = 2^{-m} \begin{pmatrix} m+1 \\ i \end{pmatrix}$$

Inoltre per questi particolari schemi si ha un analisi basata sugli autovalori che serve a vedere a cosa e se converge uno schema e, inoltre, definendo P^∞ il vettore limite del processo di suddivisione, si riesce a prevedere quale sarà il valore limite dei dati iniziali [5, 9]. Gli esempi rappresentativi di questa prima classe sono gli schemi B-spline lineari, quadratiche e cubiche.

Esempio 1.1. `B-spline lineari`

Come primo semplice esempio si considera lo schema che riproduce le funzioni B-spline lineari. La legge ad esso associata è:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \text{ con } i = 0, ..., 2^k N \\ P_{2i+1}^{k+1} = \frac{1}{2}\left(P_i^k + P_{i+1}^k\right) \text{ con } i = 0, ..., 2^k N - 1 \end{cases} \qquad k = 0, 1, 2...$$

Lo schema ha il simbolo associato:

$$a(z) = \frac{1}{2}(z+1)^2$$

quindi ha maschera della forma:

$$a = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$$

Si nota che lo schema è stazionario, uniforme e lineare per come sono i coefficienti. Questo schema è il più semplice ma non non risulta essere molto utile perché la curva limite è semplicemente la poligonale iniziale, come si può vedere dalla figura 1.1. Si ha, infatti, che lo schema riproduce le curve in C^0.

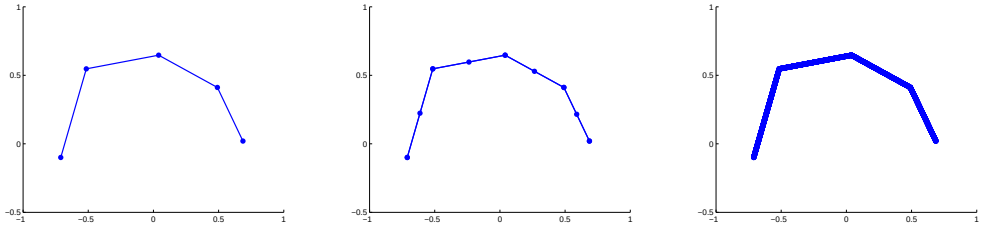


Figura 1.4: Poligonale iniziale,primo raffinamento, curva limite

Esempio 1.2. `B-spline Quadratiche`

Un altro esempio di schema di approssimazione, uniforme, stazionario e lineare è dato dalle funzioni B-spline quadratiche, chiamato anche *algoritmo di Chaikin* oppure *Corner-Cutting*. Lo schema ha la seguente legge:

$$\begin{cases} P_{2i+1}^{k+1} = \frac{1}{4}\left(P_i^k + 3P_{i+1}^k\right) \\ P_{2i}^{k+1} = \frac{1}{4}\left(3P_i^k + P_{i+1}^k\right) \end{cases}$$

Si nota che i coefficienti che definiscono lo schema sono una combinazione lineare quindi lo schema è lineare.
Questo schema converge alle B-spline quadratiche, cioè la funzione limite data dai passi di suddivisione è proprio una B-spline quadratica.
Dall'equazione 1.4, questo schema ha come simbolo associato:

$$a(z) = 2^{-2}(1+z)^3$$

quindi:

$$a(z) = \frac{1}{4}\left(1 + 3z + 3z^2 + z^3\right)$$

poi, avere questo simbolo significa avere come maschera:

$$a = \left\{\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}\right\}$$

Inoltre utilizzando l'analisi degli autovalori si riesce a stabilire quale sarà la posizione limite dei dati iniziali:

$$P_i^\infty = 0.5P_i^k + 0.5P_{i+1}^k$$

Si dice *Corner-Cutting* perché lo schema ha la proprietà di 'tagliare gli angoli', cioè, ad ogni passo di suddivisione gli angoli vengono smussati, come si nota in figura 1.5.

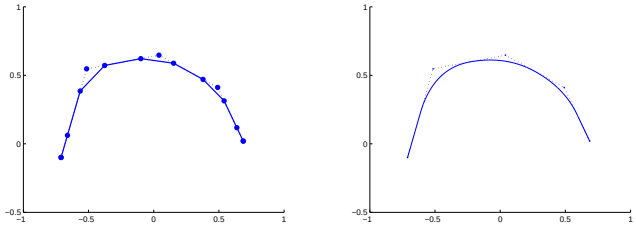


Figura 1.5: Poligonale iniziale e primo raffinamento, curva limite

Esempio 1.3. `B-spline cubiche`

Un altro interessante esempio di schema di approssimazione, uniforme, lineare e stazionario è dato dalle B-spline cubiche. In questo caso la legge è:

$$\begin{cases} P_{2i+1}^{k+1} = \frac{1}{8}\left(P_{i-1}^{k} + 6P_{i}^{k} + P_{i+1}^{k}\right) \\ P_{2i}^{k+1} = \frac{1}{8}\left(4P_{i}^{k} + 4P_{i+1}^{k}\right) \end{cases}$$

Anche questo schema converge, in particolare, alle B-spline cubiche.
Il simbolo associato a questo schema è:

$$a(z) = 2^{-3}\left(1+z\right)^4 = \frac{1}{8}\left(1 + 4z + 6z^2 + 4z^3 + z^4\right)$$

che definisce la maschera:

$$a = \left\{\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8}\right\}$$

Analizzando gli autovalori si ha che:

$$P_i^{\infty} = 0.1667P_{i-1}^{k} + 0.667P_{i}^{k} + 0.1667P_{i+1}^{k}$$

Per avere un'idea di come lavora questo schema si può osservare la seguente figura.

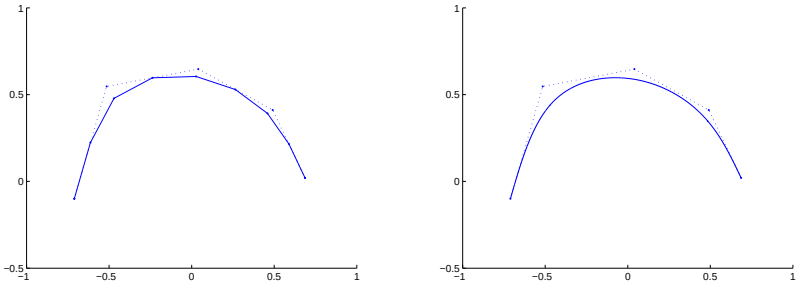


Figura 1.6: Poligonale iniziale e primo raffinamento, curva limite

1.2. **Approssimazione Uniforme Non Stazionario**

Gli schemi appartenenti a questa classe si differenziano dagli altri schemi visti perché sono non stazionari quindi esistono diverse matrici $\mathbf{S}_k$ una per ogni passo di suddivisione. Per questi schemi la maschera è della forma:

$$a^k := \{a_0^k, a_1^k, \cdots\}$$

In questo caso si hanno delle maschere diverse, una per ogni livello. Il simbolo associato a ciascuna maschera sarà:

$$a^k(z) := \sum_i a_i^k z^i \text{ con } z \in \mathbb{R} \text{ e } \forall k$$

In questa classe gli schemi sono lineari quindi, ad ogni livello, i nuovi punti si definiscono attraverso una combinazione lineare dei punti al passo precedente.
Anche per questo tipo di schemi si possono considerare separatamente i punti pari e i punti dispari [5]. Lo schema diventa quindi:

$$\begin{cases} P_{2i}^{k+1} = \sum_{j=0}^{m} a_{2i}^k P_{i+j}^k \\ P_{2i+1}^{k+1} = \sum_{j=0}^{m} a_{2i+1}^k P_{i+j}^k \end{cases} \qquad k = 0, 1, 2... \tag{1.6}$$

Gli esempi di schemi più studiati in questa classe sono quelli che riproducono le B-spline esponenziali. Ci sono diversi articoli che spiegano come si arriva alla definizione di questi schemi. La prima trattazione è di Warren e Weimer nell'articolo [13] e l'altra è di Romani in [16].
Nell'articolo di Warren e Weimer si focalizza l'attenzione sugli schemi che sono soluzioni dell'equazione differenziale non omogenea

$$\sum_{i=0}^{m} \beta_i P^{(i)}(x) = 0 \tag{1.7}$$

con β_i delle costanti reali; $P^{(i)}$ la derivata i-esima nel punto. Introducendo D_x^i, operatore differenziale all'i-esimo ordine, si può riscrivere l'equazione 1.7 come:

$$\left(\sum_{i=0}^{m} \beta_i D_x^i\right) P(x) = 0 \tag{1.8}$$

Senza perdere di generalità, l'operatore differenziale può essere fattorizzato come:

$$\sum_{i=0}^{m} \beta_i D_x^i = \prod_{i=0}^{m} (D_x - \alpha_i) \tag{1.9}$$

con α_i le radici dell'equazione polinomiale $\sum_{i=0}^{m} \beta_i x^i = 0$. Si nota che se β_i sono reali allora le α_i sono o reali oppure complesse coniugate.
Ora l'obiettivo è definire un nuovo tipo di maschera, con simbolo associato, detto *maschera delle differenze* per l'operatore differenziale 1.9 sulla griglia $2^{-k}\mathbb{Z}$ (schema uniforme).
Quindi, considerando un singolo fattore della forma $(D_x - \alpha)$ si ha che l'equazione differenziale $(D_x - \alpha_i)\, P(x) = 0$ ha soluzione $P(x) = e^{\alpha x}$. Allora, nella griglia $2^{-k}\mathbb{Z}$ si considera la sequenza geometrica

$$\left\{\ldots, \eta(k,\alpha)^{-2}, \eta(k,\alpha)^{-1}, 1, \eta(k,\alpha), \eta(k,\alpha)^2, \ldots\right\}$$

con $\eta(k,\alpha) := e^{2^{-k} * \alpha}$ e si sceglie, come simbolo delle differenze,

$$d^k(x) := (1 - \eta(k,\alpha)x)$$

Con opportune normalizzazioni (vedi [13] pp. 104) si ha che:

$$d^k(x) := \prod_{i=0}^{m} \left(\frac{\alpha}{\eta\,(k,\alpha) - 1}\right) (1 - \eta(k,\alpha)x) \tag{1.10}$$

Per la definizione dello schema, il primo passo è costruire uno schema di differenze finite della forma:

$$d^k(x)P^k(x) = 2^k d^0\left(x^{2^k}\right) P^0\left(x^{2^k}\right)$$

con d^i la maschera delle differenze all' i-esimo livello, P^0 il vettore dei punti iniziali e P^k il nuvo vettore dei punti al livello k-esimo.
Dato P^0 vettore dei punti iniziali, si ha che il vettore P^k, al livello k-esimo è dato dal simbolo:

$$a^{k-1}(x) = \frac{2d^{k-1}(x^2)}{d^k(x)}$$

che si può anche riscrivere come:

$$a^{k-1}(x) = 2\prod_{i=1}^{m} \frac{1+\eta(k,\alpha_i)x}{1+\eta(k,\alpha_i)}$$

Nell'articolo [16], invece, si cercano degli schemi che riproducono delle funzioni negli spazi

$$W_1 := \{1, x, ..., x^{n-1}, x^n, e^{tx}, e^{-tx}\}$$
$$W_2 := \{1, x, e^{tx}, e^{-tx}, ..., e^{ntx}, e^{-ntx}\} \text{ oppure}$$
$$W_3 := \{1, x, e^{tx}, e^{-tx}, ..., x^{n-1}e^{ntx}, x^{n-1}e^{-ntx}\}$$

Per gli schemi che riproducono le funzioni nello spazio W_1 si ha la seguente maschera:

$$a^k(x) = \frac{1}{2^n}(x+1)^{n+1}\frac{x^2 + 2v^{k+1}x + 1}{2(v^{k+1}+1)} \tag{1.11}$$

con $n \in \mathbb{N}$ e v^{k+1} un parametro calcolato secondo la legge di ricorrenza

$$v^{k+1} = \sqrt{\frac{1+v^k}{2}} \tag{1.12}$$

Si ha che l'equazione precedente soddisfa l'uguaglianza:

$$v^{k+1} = \frac{1}{2}(e^{\frac{t}{2^{k+1}}} + e^{-\frac{t}{2^{k+1}}})$$

quindi la maschera 1.14 si può riscrivere come:

$$a^k(x) = \frac{1}{2^n}(x+1)^{n+1}\left(\frac{e^{t_{k+1}}x + 1}{e^{t_{k+1}}+1}\right)\left(\frac{e^{-t_{k+1}}x + 1}{e^{-t_{k+1}}+1}\right) \tag{1.13}$$

con $t_{k+1} = \frac{t}{k+1}$ Come spiegato in [16], se $t = 0$, $t = s$ o $t = is$, con $s > 0$ lo schema che ha questa maschera, con $n \in \mathbb{N}$, riproduce le funzioni appartenenti allo spazio $W_1 = \{1, x, ..., x^{n-1}, x^n, e^{tx}, e^{tx}\}$. In particolare:

se $t = 0$: $W_1 = \{1, x, ..., x^{n+1}, x^{n+2}\}$ e lo schema riproduce le funzioni polinomiali B-spline in C^{n+1}; $v^0 = 1$

se $t = s$ con $s > 0$: $W_1 = \{1, x, ..., x^{n-1}, x^n, \cosh(sx), \sinh(sx)\}$ e lo schema riproduce le funzioni iperboliche B-spline in C^{n+1}; $v^0 \in (1, \infty)$

se $t = is$ con $s > 0$: $W_1 = \{1, x, ..., x^{n-1}, x^n, \cos(sx), \sin(sx)\}$ e lo schema riproduce i polinomi trigonometrici B-spline in C^{n+1}; $v^0 \in (-1, 1)$

Per gli schemi che riproducono le funzioni nello spazio W_2 si ha la seguente maschera:

$$a^k(x) = \frac{1}{2}(x+1)^2 \prod_{j=1}^{n} \frac{x^2 + \lambda_j(v^{k+1})x + 1}{(\lambda_j(v^{k+1}) + 2)} \tag{1.14}$$

con $n \in \mathbb{N}$, v^{k+1} dato dalla 1.12 e le funzioni λ_j definite come il determinante di una speciale matrice tridiagonale L la cui espressione può essere vista in [16][pp. 6 eq.(8)]. Il fatto su cui soffermarsi è che la maschera si dimostra essere equivalente a:

$$a^k(x) = \frac{1}{2}(x+1)^2 \left(\frac{e^{t_{k+1}}x + 1}{e^{t_{k+1}} + 1}\right)\left(\frac{e^{-t_{k+1}}x + 1}{e^{-t_{k+1}} + 1}\right) \cdots \left(\frac{e^{nt_{k+1}}x + 1}{e^{nt_{k+1}} + 1}\right)\left(\frac{e^{-nt_{k+1}}x + 1}{e^{-nt_{k+1}} + 1}\right) \tag{1.15}$$

con $t_{k+1} = \frac{t}{k+1}$ Anche in questo caso si possono differenziare tre situazioni diverse; se $t = 0$, $t = s$ o $t = is$, con $s > 0$ lo schema che ha questa maschera, con $n \in \mathbb{N}$, riproduce le funzioni appartenenti allo spazio $W_2 = \{1, x, e^{tx}, e^{-tx}, ..., e^{ntx}, e^{-ntx}\}$, cioè riproduce una speciale sottoclasse di funzioni L-spline in C^{2n}. In particolare:

se $t = 0$: $W_2 = \{1, x, ..., x^{2n}, x^{2n+1}\}$ e lo schema riproduce le funzioni polinomiali B-spline in C^{2n};

se $t = s$ con $s > 0$: $W_2 = \{1, x, \cosh(sx), \sinh(sx), ..., \cosh(nsx), \sinh(nsx)\}$ e lo schema riproduce le funzioni spline-in-tension in C^{2n}; $v^0 \in (1, \infty)$

se $t = is$ con $s > 0$: $W_2 = \{1, x, \cos(sx), \sin(sx), ..., \cos(nsx), \sin(nsx)\}$ e lo schema riproduce delle B-spline trigonometriche miste in C^{2n};

Infine, per gli schemi che riproducono le funzioni nello spazio W_3 si ha la seguente maschera:

$$a^k(x) = \frac{1}{2}(x+1)^2 \frac{(x^2 + 2v^{k+1}x + 1)^n}{2^n(v^{k+1} + 1)^n} \tag{1.16}$$

con $n \in \mathbb{N}$, v^{k+1} dato dalla 1.12. Anche in questo caso si ha una formulazione equivalente:

$$a^k(x) = \frac{1}{2}(x+1)^2 \frac{(e^{t_{k+1}}x + 1)^n}{(e^{t_{k+1}} + 1)^n} \frac{(e - t_{k+1}x + 1)^n}{(e^{-t_{k+1}} + 1)^n} \tag{1.17}$$

con $t_{k+1} = \frac{t}{k+1}$ Dall'articolo [16] risulta che se $t = 0$, $t = s$ o $t = is$, con $s > 0$ lo schema che ha questa maschera, con $n \in \mathbb{N}$, riproduce le funzioni appartenenti allo spazio $W_3 = \{1, x, e^{tx}, e^{-tx}, ..., x^{n-1}e^{tx}, x^{n-1}e^{-tx}\}$. In particolare:

se $t = 0$: $W_3 = \{1, x, ..., x^{2n}, x^{2n+1}\}$ e lo schema riproduce le funzioni polinomiali B-spline in C^{2n};

se $t = s$ con $s > 0$: $W_2 = \{1, x, \cosh(sx), \sinh(sx), ..., x^{n-1}\cosh(sx), x^{n-1}\sinh(sx)\}$ e lo schema riproduce le funzioni spline-in-tension in C^{2n}; $v^0 \in (1, \infty)$

se $t = is$ con $s > 0$: $W_2 = \{1, x, \cos(sx), \sin(sx), ..., x^{n-1}\cos(sx), x^{n-1}\sin(sx)\}$ e lo schema riproduce delle B-spline trigonometriche miste in C^{2n};

Esempio 1.4. `B-spline Esponenziale n=1`

Se $n = 1$ gli spazi sopra descritti risultano gli stessi.
La maschera a^k diventa:

$$a^k(x) = \frac{1}{2}(x+1)^2 \frac{(x^2 + 2v^{k+1}x + 1)}{2(v^{k+1}+1)}$$

con qualche calcolo si ha che:

$$a^k(x) = \frac{1}{4(v^{k+1}+1)}\left(x^4 + (2+2v^{k+1})x^3 + (2+4v^{k+1})x^2 + (2+2v^{k+1})x + 1\right)$$

quindi lo schema risultante è dato da:

$$\begin{cases} P^{k+1}_{2i+1} = \frac{1}{4(v^{k+1}+1)}\left(P^k_{i+1} + (2+4v^{k+1})P^k_i + P^k_{i-1}\right) \\ P^{k+1}_{2i} = \frac{1}{4(v^{k+1}+1)}\left((2+2v^{k+1})P^k_{i+1} + (2+2v^{k+1})P^k_i\right) \end{cases} \tag{1.18}$$

Si ricorda che

$$v^{k+1} = \sqrt{\frac{1+v^k}{2}}$$

quindi, lo schema così costruito ha i coefficienti dipendenti dal livello k perciò è non stazionario; inoltre è lineare perché i coefficienti sono dati da una combinazione lineare.
A seconda del valore dato a v^0, poi, si distinguono diversi casi:
Se $v^0 = (-1, 1)$ lo schema riproduce le B-spline trigonometriche in C^2
Se $v^0 = 1$ lo schema riproduce i polinomi in C^2
Se $v^0 \in (1, +\infty)$ 1 lo schema riproduce le B-spline iperboliche in C^2

1.3. **Approssimazione Non Uniforme Stazionario**

Gli schemi non uniformi sono schemi in cui esistono diverse leggi da applicare a diversi gruppi di punti. La rappresentazione parametrica dei punti non sarà più $\{(i2^{-k}, P^k_i) : i \in \mathbb{Z}\}$ ma diventerà $\{(t_i, P^k_i) : i \in \mathbb{Z}\}$
Gli schemi di questo tipo sono stazionari quindi non si cambiano le leggi a seconda dei livelli e sono lineari perché i nuovi punti si definiscono a partire da una combinazione lineare dei punti ai passi precedenti.

Esempio 1.5. `Non Uniforme Stazionario`
Un esempio di schema non uniforme stazionario è dato dalla maschera:

$$a := \left\{ \frac{\omega_i}{2}, \frac{1}{2}, (1-\omega_i), \frac{1}{2}, \frac{\omega_i}{2} \right\}$$

che diventa:

$$\begin{cases} P_{2i+1}^{k+1} = \frac{\omega_i}{2} P_{i-1}^k + (1-\omega_i) P_i^k + \frac{\omega_i}{2} P_{i+1}^k \\ P_{2i}^{k+1} = \frac{1}{2} P_i^k + \frac{1}{2} P_{i+1}^k \end{cases}$$

In cui si ha un ω_i diverso per ogni lato della poligonale iniziale. Ai passi successivi il paramentro ha forma:

$$\omega_{2i}^{k+1} = \omega_{2i+1}^{k+1} = \omega_i^k$$

come mostrato in figura.

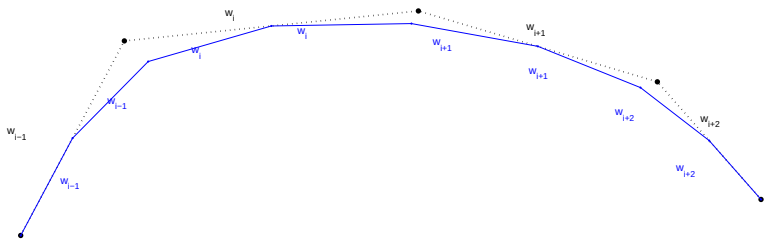


Figura 1.7: Parametro ω_i

Lo schema così definito è non uniforme perché per ogni lato la legge di suddivisione cambia. Rimane però stazionario perché la legge cambia da un lato all'altro ma non da un livello all'altro.
Se $\omega_i = 0 \forall i$ si ha che lo schema è esattamente lo schema B-spline lineare dell'esempio 1.1; se,invece, $\omega_i = \frac{1}{4} \forall i$ lo schema riproduce lo schema B-spline cubico dell'esempio 1.3. In questi due casi limite lo schema è uniforme. Per sfruttare le proprietà della non uniformità dello schema bisogna impostare un valore diverso per ogni lato della poligonale iniziale.

1.4. **Approssimazione Non Uniforme Non Stazionario**

Sabin, Sederberg, Sewell e Zhang in [14] hanno studiato tali schemi e, per la loro

definizione, hanno utilizzato il concetto di *intervallo nodale*. Si ricorda che gli schemi non uniformi hanno, come rappresentazione parametrica dei punti, $\{(t_i, P_i^k) : i \in \mathbb{Z}\}$. Si definisce allora l'intervallo nodale

$$d_i := t_{i+1} - t_i$$

Questi schemi riproducono le funzioni B-spline, allora si distinguono due casi:
Se il grado della B-spline considerata è dispari si considerano gli intervalli nodali d_i appena definiti e si avranno tanti intervalli nodali quanti sono i lati della poligonale di controllo P^0.
Se, invece, il grado della B-spline è pari si considerano i nodi t_i, un nodo per ogni vertice della poligonale di controllo P^0.
Sabin ha fornito gli esempi di B-spline quadratiche e cubiche:

Esempio 1.6. `B-spline Quadratica Non Uniforme`

In questo esempio si costruisce uno schema che riproduce le funzioni B-spline quadratiche; per le B-spline quadratiche ogni vertice della poligonale di controllo corrisponde ad un singolo segmento di curva quadratica. Si considerano, allora, i nodi della partizione nodale T e si ha il seguente schema:

$$\begin{cases} P_{2i}^{k+1} = \frac{(t_i+2t_{i+1})P_i^k + t_i P_{i+1}^k}{2(t_i+t_{i+1})} \\ P_{2i+1}^{k+1} = \frac{t_{i+1}P_i^k + (2t_i+t_{i+1})P_{i+1}^k}{2(t_i+t_{i+1})} \end{cases}$$

Se i nodi non sono equspaziati lo schema non è uniforme. Allora, per sfruttare la non uniformità dello schema bisogna impostare i nodi in modo che non siano equispaziati. Per i passi successivi al primo si utilizza la definizione semi-regolare per il posizionamento dei nuovi nodi. Significa, cioè, che i nuovi nodi sono posizionati nel punto medio dei nodi al passo precedente, in particolare si definiscono (vedi figura 1.8), come:

$$t_{2i}^{k+1} = t_i^k \text{ e } t_{2i+1}^{k+1} = \frac{t_i^k}{2}$$

Lo schema risulta essere non stazionario, infatti ad ogni livello i coefficienti cambiano di valore a seconda del valore dei nodi. I nuovi punti, inoltre sono combinazioni lineari dei punti ai passi precedenti quindi lo schema è lineare.

Esempio 1.7. `B-spline Cubica Non Uniforme`

In questo esempio si utilizzano le B-spline cubiche, si ha un grado dispari percui si utilizza la definizione di intervallo nodale e ci saranno tanti intervalli nodali quanti sono

i lati della poligonale. Lo schema ha la seguente espressione:

$$\begin{cases} P_{2i}^{k+1} = \frac{(d_i+2d_{i+1})P_i^k+(d_i+2d_{i-1})P_{i+1}^k}{2(d_{i-1}+d_i+d_{i+1})} \\ P_{2i+1}^{k+1} = \frac{d_i P_{2i-1}^{k+1}(d_{i-1}+d_i)P_i^k+d_{i-1}P_{2i+1}^{k+1}}{2(d_{i-1}+d_i+d_{i+1})} \end{cases}$$

Se gli intervalli nodali sono tutti uguali ad 1 si ha che i nodi sono equispaziati quindi lo schema risulta essere uniforme. Anche in questo caso, se si vuole sfruttare la non uniformità, bisogna definire i nodi in modo che non siano equispaziati. Ai passi successivi si utilizza il metodo semi-regolare per la definizione dei nuovi nodi. Quindi i nodi al passo $k+1$ sono posizionati nel punto medio tra i nodi del passo k, gli intervalli nodali ai passi successivi al primo, come mostrato in figura 1.8, allora possono essere definiti secondo la legge:

$$d_{2i}^{k+1} = d_{2i-1}^{k+1} = \frac{d_i^k}{2}$$

Come lo schema quadratico, anche lo schema cubico è non stazionario perché la legge dipende dal valore degli intervalli nodali ma è lineare perché i nuovi punti sono combinazione lineare dei punti al passo precedente.

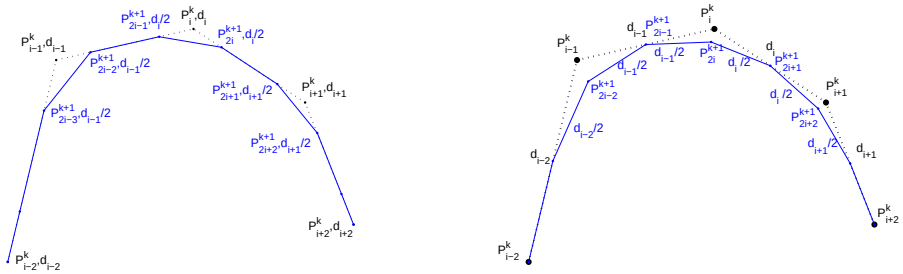


Figura 1.8: Intervalli nodali d_i per quadratiche e cubiche

1.5. **Interpolazione Uniforme Stazionario**

In questa seconda parte della classificazione si parlerà di schemi interpolanti e si ha che uno schema di interpolazione si può scrivere, in generale, come:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \sum_j \alpha_j P_{i-j}^k \text{con } i \in \mathbb{Z}, k \in \mathbb{Z}^+ \end{cases} \tag{1.19}$$

Gli schemi appartenenti a questa classe sono stazionari quindi esiste un'unica maschera che risulta essere della forma:

$$a^k := \left\{a^k_{-L}, 0, a^k_{-L+1}, 0, ..., a^k_{-1}, 1, a^k_0, 0, a^k_1, 0, ..., 0, a^k_U\right\}$$

Si nota che si possono considerare separatamente i coefficienti relativi ai punti pari e a quelli dispari:

$$\begin{cases} a^k_0 = 1 \\ a^k_{2i} = 0 \\ a^k_{2i+1} = a_j \text{ con } j = -L, -L+1, ..., U \end{cases}$$

Inoltre se lo lo schema converge allora valgono le seguenti equazioni [5, 8]:

$$a^k(1) = 2 \text{ e } a^k(-1) = 0 \ \forall k \tag{1.20}$$

Di seguito è riportato un semplice esempio di schema interpolante; si procede poi con la teoria di Dubuc e Deslauries che è alla base della definizione degli schemi di interpolazioni; infine sono riportati diversi esempi di schemi a n punti.

Esempio 1.8. `Schema a 2 punti`

Un primo semplice esempio di questo tipo di schemi è l'interpolazione lineare tra i due punti finali di un intervallo che prende il loro valore intermedio. La legge ad esso associata è:

$$P_{i+\frac{1}{2}} = \frac{1}{2}\left(P_i + P_{i+1}\right),\ i = 0, ..., N$$

quindi le iterazioni diventano:

$$\begin{cases} P^{k+1}_{2i} = P^k_i & i = 0, ..., 2^k N \\ P^{k+1}_{2i+1} = \frac{1}{2}\left(P^k_i + P^k_{i+1}\right) & i = 0, ..., 2^k N - 1 \end{cases} \quad k = 0, 1, 2...$$

La maschera relativa allo schema ha forma:

$$a = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$$

Lo schema ha, come simbolo associato:

$$a(z) = \frac{1}{2}\left(z + 1\right)^2$$

Si nota che lo schema è stazionario, uniforme e lineare per come sono i coefficienti. Questo schema di suddivisione riproduce i polinomi di grado 1 ed è esattamente lo stesso schema dell'esempio 1.1 per le funzioni B-spline lineari.

Tornando alla teoria degli schemi, Deslauries e Dubuc, in [3], hanno studiato gli schemi di suddivisione interpolanti. L'idea si basa sulla costruzione di un procedimento per induzione che estende una funzione discreta in una funzione continua. Partendo da $y(n)$, una sequenza di punti pensati sull'insieme $\mathbb{Z}$, si vuole trovare una sua estensione $y(t)$ in tutto l'asse reale. Questo procedimento sarà uno schema di interpolazione simmetrico iterativo. Comunque il risultato più importante è che: l'estensione di $y(n)$, su $\mathbb{R}$, è

$$y(t) := \sum_n y(n)F(t-n)$$

Fissando i parametri: $2N$ il numero di punti di controllo iniziali e n un numero intero che indica il livello di suddivisione si ha che:

$$F\left(n+\frac{1}{2}\right) = L_{-n}\left(\frac{1}{2}\right)$$

con $\{L_k\}_{k\in S}$ i polinomi di Lagrange.
Se $S = \{-N+1, -N+2, ..., N-1, N\}$, cioè i punti sono simmetrici, si parla di *interpolazione simmetrica di Lagrange*.

Esempio 1.9. `N=2`
Se $N = 2$ si ha che $S = \{-1, 0, 1, 2\}$; siccome $F\left(n+\frac{1}{2}\right) = L_{-n}\left(\frac{1}{2}\right)$, si ha bisogno dei polinomi di Lagrange relativi a quella sequenza, che sono:

$$\begin{gathered} L_{-1}(x) = -x(1-x)(2-x)/6 \\ L_0(x) = (x+1)(1-x)(2-x)/2 \\ L_1(x) = (x+1)x(2-x)/2 \\ L_2(x) = -(x+1)x(1-x)/6 \end{gathered}$$

In più, siccome F è simmetrica, si riportano solo i primi due valori:

$$\begin{gathered} F\left(0+\tfrac{1}{2}\right) = L_0\left(\tfrac{1}{2}\right) \\ F\left(1+\tfrac{1}{2}\right) = L_{-1}\left(\tfrac{1}{2}\right) \end{gathered}$$

quindi:

$$\begin{array}{rrrr} F\left(\frac{1}{2}\right) = & L_0\left(\frac{1}{2}\right) = & (\frac{1}{2}+1)(1-\frac{1}{2})(2-\frac{1}{2})/2 = & \frac{9}{16} \\ F\left(\frac{3}{2}\right) = & L_{-1}\left(\frac{1}{2}\right) = & -\frac{1}{2}(1-\frac{1}{2})(2-\frac{1}{2})/6 = & \frac{1}{16} \end{array}$$

Quindi:

t	$\frac{1}{2}$	$\frac{3}{2}$
$F(t)$	$\frac{9}{16}$	$-\frac{1}{16}$

Esempio 1.10. `N=3`
Se, invece, $N = 3$ si ha $S = \{-2, -1, 0, 1, 2, 3\}$ e i polinomi di Lagrange relativi a questa sequenza sono:

$$\begin{gathered} L_{-2} = (x+1)x(1-x)(2-x)(3-x)/120 \\ L_{-1}(x) = -(x+2)x(1-x)(2-x)(3-x)/24 \\ L_0(x) = (x+2)(x+1)(1-x)(2-x)(3-x)/12 \\ L_1(x) = (x+2)(x+1)x(2-x)(3-x)/12 \\ L_2(x) = -(x+2)(x+1)x(1-x)(3-x)/24 \\ L_3(x) = (x+2)(x+1)x(1-x)(2-x)/120 \end{gathered}$$

Si ha che $F\left(n+\frac{1}{2}\right) = L_{-n}\left(\frac{1}{2}\right)$, per cui:

$$\begin{array}{ll} F\left(0+\frac{1}{2}\right) = F\left(\frac{1}{2}\right) & = L_0\left(\frac{1}{2}\right) = \frac{75}{128} \\ F\left(1+\frac{1}{2}\right) = F\left(\frac{3}{2}\right) & = L_{-1}\left(\frac{1}{2}\right) = -\frac{25}{256} \\ F\left(2+\frac{1}{2}\right) = F\left(\frac{5}{2}\right) & = L_{-2}\left(\frac{1}{2}\right) = \frac{3}{256} \end{array}$$

Quindi:

t	1/2	3/2	5/2
$F(t)$	75/128	−25/256	3/256

Osservazione 1. Gli schemi di Dubuc-Deslauries con $2N$ punti iniziali riproducono i polinomi di grado $2N - 1$.

Da queste osservazioni si ricavano le leggi per gli schemi di interpolazione. Infatti tali schemi sono interpolanti perché sono state utilizzate le valutazioni nei punti del polinomio interpolante di Lagrange.
Gli schemi sono uniformi perché le valutazioni sono state fatte nel valore intermedio tra due punti, cioè in $\frac{1}{2}$.
Il numero $2N$ indica quanti punti del livello k si utilizzano per determinare un nuovo punto al livello $k+1$. Negli esempi si utilizzano rispettivamente 4 e 6 punti.
Praticamente si utilizzano le valutazioni della funzione F per scrivere le maschere nel seguente modo, senza considerare i coefficienti relativi ai punti pari:

- Nell'esempio 1.9 la maschera è

$$\overline{a} := \left\{F\left(\frac{3}{2}\right), F\left(\frac{1}{2}\right), F\left(\frac{1}{2}\right), F\left(\frac{3}{2}\right)\right\} = \left\{-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16}\right\}$$

- Nell'esempio 1.10 la maschera è

$$\begin{gathered} \overline{a} := \left\{F\left(\frac{5}{2}\right), F\left(\frac{3}{2}\right), F\left(\frac{1}{2}\right), F\left(\frac{1}{2}\right), F\left(\frac{3}{2}\right), F\left(\frac{5}{2}\right)\right\} = \\ = \left\{\frac{3}{256}, -\frac{25}{256}, \frac{75}{128}, \frac{75}{128}, -\frac{25}{256}, \frac{3}{256}\right\} \end{gathered}$$

Esempio 1.11. `Schema a 4 punti`

In questo esempio si vede l'applicazione delle osservazioni di Dubuc e Deslauries relative all'esempio 1.9. In questo schema, per trovare il nuovo punto, si utilizzano quattro punti del livello precedente. La maschera relativa è

$$a = \left\{ -\frac{1}{16}, 0, \frac{9}{16}, 1, \frac{9}{16}, 0, \frac{1}{16} \right\}$$

e lo schema diventa:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{9}{16}\left(P_i^k + P_{i+1}^k\right) - \frac{1}{16}\left(P_{i-1}^k + P_{i+2}^k\right) \end{cases}$$

Lo schema, oltre ad essere stazionario e uniforme, è lineare perché i nuovi punti sono combinazione lineare dei punti al livello precedente, [5, 9].
Lo schena è denominato *Four Point* perché per determinare un nuovo punto P_{2i+1}^{k+1} si ha bisogno di quattro punti del livello precedente, $P_{i-1}^k, P_i^k, P_{i+1}^k$ e P_{i+2}^k; come mostrato in figura 1.9.

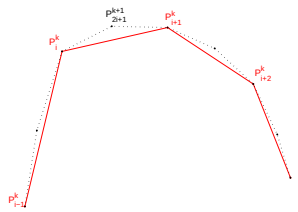


Figura 1.9: Four Point

Inoltre si può osservare che il simbolo relativo a questo schema è:

$$a(z) = \frac{1}{2z}(z+1)^2\left(1 + \frac{1}{16}b(z)\right) \text{ con } b(z) = -2z^{-2}(z-1)^2(z^2+1)$$

Osservazione 2. Un altro modo per arrivare allo stesso risultato è suggerito da N. Dyn in [8]. Uno schema di interpolazione a quattro punti sarà della forma:

$$P_{2i+1}^{k+1} = a_1\left(P_i^k + P_{i+1}^k\right) + a_2\left(P_{i-1}^k + P_{i+2}^k\right)$$

Si interpolano i punti di controllo con un polinomio cubico $q(x) \in P[x^3]$ tale che:

$$q_i(j) = P_j^k, j = i-1, i, i+1, i+2$$

perció si riesce a calcolare

$$P_{2i+1}^{k+1} = q_i \left(1 + \frac{1}{2}\right)$$

Poi si pone $f_i^k = (i2^k)^n$ ma si vuole uno schema stazionario percui k non interessa $f_i = (i)^n$
Per x^0, $n = 0$ e $i = 0$ quindi $f_i = 1$: $P_1^{k+1} = 1 == a_1(1+1) + a_2(1+1)$
Per x^2 $n = 2$ e $i = 0$ quindi $f_i = i^2$: $P_1^{k+1} = \frac{1}{4} == a_1(0+1) + a_2(1+4)$
Non si calcola per $n = 1, 3$ perché i polinomi sono invarianti per traslazioni Quindi:

$$\begin{cases} 1 = 2a_1 + 2a_2 \\ 1 = 4a_1 + 20a_2 \end{cases} \Rightarrow \begin{cases} a_1 = 9/16 \\ a_2 = -1/16 \end{cases}$$

quindi si ritrova lo stesso risultato di prima.

Osservazione 3. *Lo schema riproduce i polinomi cubici.*
Infatti, grazie alla linearità dello schema, osservare che riproduca i polinomi di terzo grado è equivalente a vedere se riproduce i monomi x^i con $i = 0, 1, 2, 3$. Ci si chiede se:

$$P_{2i+1}^{k+1} = x^i$$

- x^0 cioè le costanti:
  Sia $P_i^k = 1\ \forall i \in \mathbb{Z}$ allora dallo schema si ha che:

  $$9/16(1+1) - 1/16(1+1) = 1$$

  Quindi

  $$P_{2i+1}^{k+1} = 1$$

- x
  Sia $P_i^k = (i2^k)\ \forall i \in \mathbb{Z}$ allora dallo schema si ha che:

  $$P_1^{k+1} = 9/16(0 + 2^{-k}) - 1/16[-2^{-k} + 2(2^{-k})) = 2^{-(k+1)}$$

- x^2
  Sia $P_i^k = (i2^k)^2\ \forall i \in \mathbb{Z}$ allora:

  $$P_1^{k+1} = 9/16(0 + (2^{-k})^2) - 1/16[-2^{-2k} + 4(2^{-2k})) = (2^{-(k+1)})^2$$

- x^3
  Sia $P_i^k = (i2^k)^3\ \forall i \in \mathbb{Z}$ quindi:

  $$P_1^{k+1} = 9/16(2^{-3k}) - 1/16[-2^{-3k} + 8(2^{-3k})) = (2^{-(k+1)})^3$$

Dalle osservazioni precedenti si deduce che lo schema quindi riproduce i polinomi cubici. ⊡

Esempio 1.12. `Schema a 6 punti`

Questo schema è un'altra applicazione degli schemi di Dubuc e Deslauries. Si fa riferimento all'esempio 1.10 e si ha che, per trovare il nuovo punto, si utilizzano sei punti del livello precedente. La maschera relativa a tutti i punti è

$$a = \left\{ \frac{3}{256}, 0, -\frac{25}{256}, 0, \frac{75}{128}, 1, \frac{75}{128}, 0, -\frac{25}{256}, 0, \frac{3}{256} \right\}$$

Quindi si ha lo schema:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{75}{128}\left(P_i^k + P_{i+1}^k\right) - \frac{25}{256}\left(P_{i-1}^k + P_{i+2}^k\right) + \frac{3}{256}\left(P_{i-2}^k + P_{i+3}^k\right) \end{cases}$$

Anche questo schema è uniforme, stazionario e lineare per come sono i valori dei coefficienti.
Poi si ha che lo schema riproduce i polinomi di grado 5 [8].

Osservazione 4. In analogia con l'osservazione dell'esempio 1.11 si vede che lo schema può essere costruito nel modo indicato da Dyn in [5].
Si cerca una maschera del tipo $a = \{\alpha_j, j = \pm 3, \pm 2, \pm 1\}$, richiedendo che sia esatto per x^i con $i = 0, 1, 2, 3, 4, 5$. Siccome i polinomi sono invarianti per traslazioni è sufficiente considerare l'intersezione nel punto $\frac{1}{2}$ basata sui valori nei punti -2,-1,0,1,2,3. Si procede come l'esempio precedente sapendo che uno schema a 6 punti sarà della forma:

$$P_{2i+1}^{k+1} = a_1\left(P_i^k + P_{i+1}^k\right) + a_2\left(P_{i-1}^k + P_{i+2}^k\right) + a_3\left(P_{i-2}^k + P_{i+3}^k\right)$$

Come sopra si pone $f_i = (i)^n$
Per x^0, $n = 0$ e $i = 0$ quindi $f_i = 1$:
$P_1^{k+1} = 1 == a_1(1+1) + a_2(1+1) + a_3(1+1)$
Per x^2 $n = 2$ e $i = 0$ quindi $f_i = i^2$:
$P_1^{k+1} = \frac{1}{4} == a_1(0+1) + a_2(1+4) + a_3(4+9)$
Per x^4 $n = 4$ e $i = 0$ quindi $f_i = i^4$:
$P_1^{k+1} = \frac{1}{16} == a_1(0+1) + a_2(1+16) + a_3(16+81)$ Lo stesso non si calcola per $n = 1, 3, 5$ perché i polinomi sono invarianti per traslazioni Quindi:

$$\begin{cases} 1 = 2a_1 + 2a_2 + 2a_3 \\ 1/4 = a_1 + 5a_2 + 13a_3 \\ 1/16 = a_1 + 17a_2 + 97a_3 \end{cases} \Rightarrow \begin{cases} a_1 = 75/128 \\ a_2 = -25/256 \\ a_3 = 3/256 \end{cases}$$

Perciò si ritrova esattamente lo schema di prima.⊡

Esempio 1.13. `Schema a` n `punti`

Con la tecnica di Dubuc-Deslauries si possono considerare anche schemi che utilizzano più punti e che riproducono polinomi con diverso grado di regolarità. Dall'osservazione 1 si ha infatti che se lo schema è definito da $2n$ punti allora riproduce polinomi di grado $2n-1$. Di seguito si riassumono gli schemi già visti nei precedenti esempi e si riportano altri schemi a n punti. Le maschere riportate si riferiscono ai soli punti dispari perché, in quanto schemi di approssimazione, per i punti pari i valori della maschera sono tutti uguali a 1.

Schema 2 Punti $\Rightarrow a = (0.5, 0.5)$ e riproduce i polinomi lineari;

Schema 4 Punti $\Rightarrow a = (-0.0625, 0.5625, 0.5625, -0.0625)$ e riproduce i polinomi di grado 3

Schema 6 Punti $\Rightarrow a = (0.0117, -0.0977, 0.5859, 0.5859, -0.0977, 0.0117)$ e riproduce i polinomi di grado 5

Schema 8 Punti $\Rightarrow a = (-0.0024, 0.0239, -0.1196, 0.5981, 0.5981, -0.1196, 0.0239, -0.0024)$ e riproduce i polinomi di grado 7

Schema 10 Punti $\Rightarrow a = (0.0005, -0.0062, 0.0346, -0.1346, 0.6056, 0.6056, -0.1346, 0.0346, -0.0062, 0.0005)$ e riproduce i polinomi di grado 9

Schema 12 Punti $\Rightarrow a = (-0.0001, 0.0016, -0.0104, 0.0436, 0.1454, 0.6107, 0.6107, -0.1454, 0.0436, -0.0104, 0.0016, -0.0001)$ e riproduce i polinomi di grado 11

Esistono anche schemi di interpolazione lineare, uniformi e stazionari che dipendono da un parametro. Di seguito sono presenti due esempi

Esempio 1.14. `Schema a 4 punti con parametro`

Esistono anche degli schemi che non hanno una formulazione espicita ma che dipendono da un parametro. Questi schemi nascono dall'idea di avere uno schema che si combinazione convessa di due schemi espliciti. Si deve la formulazione di questo schema a N.Dyn ed a D. Levin, [5].
In questo esempio si vuole trovare uno schema a quattro punti che sia combinazione convessa degli schemi degli esempi 1.8 e 1.11; che hanno leggi: (si è interessati solo ai punti dispari)

$$\begin{array}{ll} \text{2punti} \rightarrow & P_{2i+1}^{k+1} = \frac{1}{2}\left(P_i^k + P_{i+1}^k\right) \\ \text{4punti} \rightarrow & P_{2i+1}^{k+1} = \frac{9}{16}\left(P_i^k + P_{i+1}^k\right) - \frac{1}{16}\left(P_{i-1}^k + P_{i+2}^k\right) \end{array}$$

Allora, sia $\omega \in \mathbb{R}$ tale che $0 \leq \omega \leq \frac{1}{16}$, i coefficienti della nuova maschera saranno:

$$\begin{cases} a_1 = \frac{1}{2}(1 - 16\omega) + \frac{9}{16}(16\omega) \\ a_2 = -\frac{1}{16}(16\omega) \end{cases} \Rightarrow \begin{cases} a_1 = \frac{1}{2} + \omega \\ a_2 = -\omega \end{cases}$$

Quindi lo schema diventa:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \ , \ i \in \mathbb{Z} \\ P_{2i+1}^{k+1} = -\omega \left(P_i^k + P_{i+1}^k\right) + \left(\frac{1}{2} + \omega\right) \left(P_{i-1}^k + P_{i+2}^k\right) \ , \ i \in \mathbb{Z} \end{cases} \qquad k = 0, 1, 2...$$

La maschera è: $a = \left(-\omega, 0, \frac{1}{2} + \omega, 1, \frac{1}{2} + \omega, 0, -\omega\right)$
e il simbolo generato:

$$a_\omega(z) = \frac{1}{2z}(z+1)^2 \left(1 + -\omega 2 z^{-2}(z-1)^2(z^2+1)\right)$$

La scelta del parametro ω_i è molto importante perché potrebbero essere calcolati risultati molto diversi tra loro, Figura 1.10

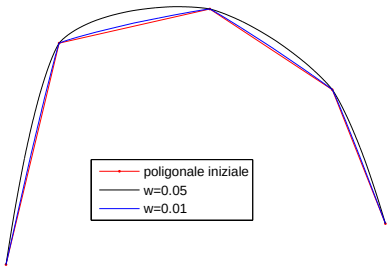


Figura 1.10: Four point con parametro

Inoltre lo schema genera una funzione limite continua solo se $|\omega| < \frac{1}{4}$, una funzione limite C^1 solo se $0 < \omega < \frac{1}{8}$. Con $\omega = 0$ si ha l'esempio 1.8 e con $\omega = \frac{1}{16}$ l'esempio 1.11.
Se $\omega \neq \frac{1}{16}$ lo schema riproduce i polinomi lineari, ma non i polinomi quadratici.
Se $\omega = \frac{1}{16}$, invece, lo schema riproduce i polinomi cubici.
Lo schema generato è lineare, stazionario ed uniforme perché combinazione di due schemi con le stesse proprietà.

Esempio 1.15. `Schema a 6 punti con parametro`

Ora si vuole costruire uno schema visto come combinazione convessa partendo dagli schemi di Dubuc-Deslauries a 4 ed a 6 punti:

$$\begin{array}{ll} \text{4punti} \rightarrow & P_{2i+1}^{k+1} = \frac{9}{16}\left(P_i^k + P_{i+1}^k\right) - \frac{1}{16}\left(P_{i-1}^k + P_{i-2}^k\right) \\ \text{6punti} \rightarrow & P_{2i+1}^{k+1} = \frac{75}{128}\left(P_i^k + P_{i+1}^k\right) - \frac{25}{256}\left(P_{i-1}^k + P_{i+2}^k\right) + \frac{3}{256}\left(P_{i-2}^k + P_{i+3}^k\right) \end{array}$$

Per costruire il nuovo schema si combinano i coefficienti della maschera in modo convesso. Sia $\omega \in \mathbb{R}$ tale che $0 \leq \omega \leq 0$ allora i coefficienti della nuova maschera saranno:

$$\begin{cases} a_1 = \frac{9}{16}(1-\omega) + \frac{75}{128}\omega \\ a_2 = -\frac{1}{16}(1-\omega) - \frac{25}{256}\omega \\ a_3 = \frac{3}{256}\omega \end{cases} \Rightarrow \begin{cases} a_1 = \frac{9}{16} + \frac{3}{128}\omega \\ a_2 = -\frac{1}{16} - \frac{9}{256}\omega \\ a_3 = \frac{3}{256}\omega \end{cases}$$

Sia $\theta = \frac{3}{256}\omega$ allora si ottiene lo schema:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \left(\frac{9}{16} + 2\theta\right)\left(P_i^k + P_{i+1}^k\right) - \left(\frac{1}{16} + 3\theta\right)\left(P_{i-1}^k + P_{i+2}^k\right) + \theta\left(P_{i-2}^k + P_{i+3}^k\right) \end{cases}$$

Se $\omega = 0$ è esattamente lo schema a 4 punti.
Se $\theta \neq \frac{3}{256}$, quindi $\omega \neq 1$, lo schema riproduce i polinomi cubici.
Se $\theta = \frac{3}{256}$, quindi $\omega = 1$, si ha esattamente lo schema a 6 punti [8].

1.6. **Interpolazione Uniforme Non Stazionario**

Questa sezione riguarda gli schemi di interpolazione non stazionari quindi la maschera cambia da passo a passo. Sono comuni, in questa classe, gli schemi che riproducono delle coniche o delle sezioni di esse.
Diversi tipi di schemi stati studiati da Casciola, Beccari e Romani in [17], da Dyn e Levin in [8], e da Kobbelt in [2, 11, 12]. Di seguito sono spiegati tutti i tre tipi separatamente.

Esempio 1.16. `Schema a 4 punti non stazionario`

Nel caso dello schema presentato da Casciola, Beccari e Romani in [17], si costruisce uno schema a quattro punti che rappresenti gli elementi degli spazi lineari V_0, V_t e V_{it} generati rispettivamente dalle funzioni $\{1, x, x^2, x^3\}$, $\{1, x, e^{tx}, e^{-tx}\}$ e $\{1, x, e^{tix}, e^{-tix}\}$. Considerando l'equazione differenziale $D^4 \cdot -t^4 D^2 \cdot = 0$ si nota che ha come soluzioni delle combinazioni lineari degli spazi V_0, V_t con $t > 0$ e V_{it} con $it > 0$. A seconda del valore del parametro di tensione t lo schema riprodurrà diverse coniche o sezioni di esse. In particolare:

- $t = 0$ si avranno dei polinomi cubici
- $t = s$ con $s > 0$ si avranno delle funzioni iperboliche
- $t = is$ con $s > 0$ si avranno delle funzioni trigonometriche

Si cercano allora delle funzioni che interpolino uniformemente i dati iniziali scritti come $\left(2^{-k}h; p^k_{j+h}\right)$, con $h = -1; 0; 1; 2$, con una funzione della forma $f(x) = a_0 + a_1x + a_2e^{tx} + a_3e^{-tx}$. Si ottiene allora:

$$\begin{cases} f(-\frac{1}{2k}) = p^k_{j-1} \\ f(0) = p^k_j \\ f(\frac{1}{2k}) = p^k_{j+1} \\ f(\frac{1}{2k-1}) = p^k_{j+2} \end{cases}$$

Da cui, dopo diversi passaggi spiegati in [17] si trova il seguente schema:

$$\begin{cases} p^{k+1}_{2j} = p^k_j \\ p^{k+1}_{2j+1} = \left(\frac{1}{2} + \omega_{k+1}\right)\left(p^k_j + p^k_{j+1}\right) - \omega_{k+1}\left(p^k_{j-1} + p^k_{j+2}\right) \end{cases}$$

con

$$\omega^{k+1} = \frac{1}{8v^{k+1}\left(1 + v^{k+1}\right)}$$

I valori del parametri v^k rispettano la seguente legge:

$$v^{k+1} = \sqrt{\frac{1 + v^k}{2}}$$

quindi a seconda del valore di v^0 si avranno diversi risultati. In particolare:

- se $v^0 = 1$ allora $v^k = 1 \forall k > 0$ e $\omega^k = \frac{1}{16} \forall k \geq 1$; lo schema diventa lo schema a 4 punti classico dell'esempio 1.11 che, quindi, riproduce i polinomi cubici.
- se $v^0 \in (1, \infty)$ allora $\omega^k = \frac{-1}{16\cosh(\frac{s}{2^k})\cosh^2(\frac{s}{2^{k+1}})}$ e lo schema riproduce le funzioni iperboliche nello spazio V_t
- se $v^0 \in (-1, 1)$ allora $\omega^k = \frac{-1}{16\cos(\frac{s}{2^k})\cos^2(\frac{s}{2^{k+1}})}$ e lo schema riproduce le funzioni trigonometriche nello spazio V_it

Lo schema è lineare, uniforme perché i nuovi punti sono una combinazione lineare dei punti ai passi precedenti e per come è stata scelta la parametrizzazione. Inoltre, se $v^0 \neq 0$ si ha che lo schema non è stazionario perché i coefficienti dipendono dal livello k considerato.

Esempio 1.17. `Schema a 4 punti interpolante un cerchio`

Nella teoria di Dyn e Levin, in [8], la legge di raffinamento riproduce le funzioni esponenziali del tipo $\{1, exp(it), exp(-it)\}$.
Si osserva, preliminarmente, che il cerchio, con centro (x_0, y_0) e raggio r, può essere rappresentato parametricamente come

$$\begin{cases} x(t) = x_0 + r\cos t \\ y(t) = y_0 + r\sin t \end{cases}$$

Inoltre vale la relazione

$$span\{1, exp(it), exp(-it)\} \supseteq span\{1, \cos t, \sin t\} \tag{1.21}$$

In questo caso si vogliono utilizzare 4 punti per costruire uno schema esatto per ogni funzione nello spazio generato da $\{1, exp(t), exp(it), exp(-it)\}$. Con la relazione 1.21, si può considerare lo spazio $\{1, t, \cos t, \sin t\}$
Si sa che uno schema a 4 punti è della forma:

$$P_{2i+1}^{k+1} = a_{-1}^k P_{i-1}^k + a_0^k P_i^k + a_1^k P_{i+1}^k + a_2^k P_{i+2}^k$$

Si assume che il punto di inserimento sia $P_{2i+1}^{k+1} = \theta 2^{-k-1}$ e che:

$$\begin{cases} P_{i-1}^k = -\theta 2^{-k} \\ P_i^k = 0 \\ P_{i+1}^k = \theta 2^{-k} \\ P_{i+2}^k = -2\theta 2^{-k} \end{cases}$$

Quindi:

$$\begin{cases} 1 \Rightarrow 1 = a_{-1}^k + a_0^k + a_1^k + a_2^k \\ t \Rightarrow \theta 2^{-k-1} = -a_{-1}^k \theta 2^{-k} + a_1^k \theta 2^{-k} - a_2^k 2\theta 2^{-k} \\ \cos t \Rightarrow \cos\theta 2^{-k-1} = a_{-1}^k cos\theta 2^{-k} + a_0^k + a_1^k \cos\theta 2^{-k} + a_2^k \cos 2\theta 2^{-k} \\ \sin t \Rightarrow \sin\theta 2^{-k-1} = a_{-1}^k \sin\theta 2^{-k} + a_1^k \sin\theta 2^{-k} + a_2^k \sin 2\theta 2^{-k} \end{cases}$$

La soluzione di questo sistema è data da:

$$\begin{cases} a_0^k = a_1^k = \frac{-1}{16\cos^2(\theta 2^{-k-2})\cos(\theta 2^{-k-1})} \\ a_{-1}^k = a_2^k = \frac{\left(1+2\cos(\theta 2^{-k-1})\right)^2}{16\cos^2(\theta 2^{-k-2})\cos(\theta 2^{-k-1})} \end{cases}$$

Quindi lo schema risultante è

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{\left(1+2\cos(\theta 2^{-k-1})\right)^2}{16\cos^2(\theta 2^{-k-2})\cos(\theta 2^{-k-1})}\left(P_{i-1}^k + P_{i+2}^k\right) + \frac{-1}{16\cos^2(\theta 2^{-k-2})\cos(\theta 2^{-k-1})}\left(P_i^k + P_{i+1}^k\right) \end{cases}$$

Anche in questo caso si nota che lo schema è lineare, uniforme ma non stazionario.
Si nota che se $k \to \infty$ allora lo schema diventa quello a 4 punti dell'esempio 1.11
Quando si applica la legge di raffinamento a punti equidistribuiti
$\left\{P_j^0, j = 0, ..., N\right\}$ sull'arco di cerchio di angolo $N\theta$, si ricopre l'arco di cerchio attraverso i punti P_j^0, $j = 2, ..., N-2$.
Se i punti sono equidistribuiti sul cerchio allora la legge di raffinamento definisce una funzione che ricopre l'intero cerchio. Per questo obiettivo bisogna definire i punti in maniera periodica, cioè se $N_k := 2^k(N+1) - 1$ allora, per ogni livello di raffinamento si definiscono i punti al bordo come:

$$\begin{cases} P_{-1}^k = P_{N_k}^k \\ P_{-2}^k = P_{N_k-2}^k \\ P_{N_k+1}^k = P_{j-1}^k \\ P_{N_k+2}^k = P_{j-2}^k \end{cases}$$

Un altro esempio di schema uniforme ma non stazionaro è stato introdotto da Leif Kobbelt in [2, 11, 12]. Egli ha studiato una nuova classe di raffinamenti interpolanti in cui, ad ogni passo di raffinamento, il nuovo punto è determinato risolvendo un problema di ottimizzazione. Scegliendo delle funzioni appropriate da minimizzare, ad ogni passo, si possono produrre delle funzioni limite molto regolari.
Si cerca uno schema di suddivisione tale che, data la poligonale $P^m = (P_0^m, ..., P_{n-1}^m)$, la poligonale raffinata $P^{m+1} = (P_0^{m+1}, ..., P_{2n-1}^{m+1})$ sia più regolare possibile. Per essere in grado di confrontare la regolarità di una poligonale si introduce il concetto di *funzionale energetico* $E(P^{m+1})$ che misura il totale di energia discreta di P^{m+1}. La legge di raffinamento al passo k è scelta in modo tale che il funzionale sia minimo.
Per semplificare la descrizione dello schema si suppone di avere delle poligonali *chiuse*.
Si introduce il concetto di *curvatura* di una poligonale, definendo la misura locale della regolarità di un sottoinsieme di vertici P_i^{m+1}, cioè:

$$K(P_i^{m+1}) := \sum_{j=0}^{k} a_j P_{i+j-r}^{m+1}$$

si introduce r in modo che $K(P_i^{m+1})$ dipenda da $P_{i-r}^{m+1}, ..., P_{i+k-r}^{m+1}$. Ogni misura di K é associata al polinomio caratteristico $\alpha z = \sum_{j=0}^{k} a_j z^j$.
Si definisce il *funzionale energetico* come:

$$E(P^{m+1}) := \sum_{i=0}^{2n-1} K(P_i^{m+1})^2 \tag{1.22}$$

L'obiettivo è minimizzare tale funzionale in modo da ridurre al minimo i salti della funzione limite. Se ad ogni passo di raffinamento si riducono al minimo i salti della poligonale, la funzione limite sarà molto più regolare rispetto alla funzione limite generata con uno schema classico.
Siccome si lavora con schemi di interpolazione si ha che i punti con indice pari sono fissati; per cui si lavora solo con i punti con indice dispari P^m_{2i+1} che sono parametri liberi del problema di ottimizzazione. Per trovare il minimo del funzionale, si calcola la derivata e si cercano le radici.

$$\begin{aligned}\frac{\partial}{\partial P^{m+1}_{2l+1}} E(P^{m+1}) &= \sum_{i=0}^{k} \frac{\partial}{\partial P^{m+1}_{2l+1}} K(P^{m+1}_{2l+1+r-i})^2 = \\ &= 2\sum_{i=0}^{k} a_i \sum_{j=0}^{k} a_j P^{m+1}_{2l+1-i+j} = \\ &= 2\sum_{i=-k}^{k} \beta_i P^{m+1}_{2l+1+i}\end{aligned} \tag{1.23}$$

con i coefficienti

$$\beta_{-i} = \beta_i = \sum_{j=0}^{k-i} a_j a_{j+1},\ i = 0, ..., k \tag{1.24}$$

Quindi, separando i punti fissi $P^{m+1}_{2i} = P^m_i$ dalle variabili il funzionale energetico diventa minimo se i nuovi punti P^{m+1}_{2i+1} sono soluzioni del sistema lineare:

$$\begin{aligned}&\begin{pmatrix} \beta_0 & \beta_2 & \beta_4 & \dots & \beta_2 \\ \beta_2 & \beta_0 & \beta_2 & \dots & \beta_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_2 & \beta_4 & \beta_6 & \dots & \beta_0 \end{pmatrix} \begin{pmatrix} P^{m+1}_1 \\ P^{m+1}_3 \\ \vdots \\ P^{m+1}_{2n-1} \end{pmatrix} = \\ &= \begin{pmatrix} -\beta_1 & -\beta_1 & -\beta_3 & \dots & -\beta_3 \\ -\beta_3 & -\beta_1 & -\beta_1 & \dots & -\beta_5 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\beta_3 & -\beta_5 & -\beta_7 & \dots & -\beta_1 \end{pmatrix} \begin{pmatrix} P^m_0 \\ P^m_1 \\ \vdots \\ P^m_{n-1} \end{pmatrix}\end{aligned} \tag{1.25}$$

Entambe queste matrici sono simmetriche e circolanti. L'analogia tra il fairing di una curva e il raffinamento con il metodo variazionale è data dal'equazione

$$\sum_{i=-k}^{k} \beta_i P^{m+1}_{2l+1+i} = 0 \tag{1.26}$$

che è chiamata *Equazione di Eulero-Lagrange*

In [12] si vede che definendo la matrice $C := [\beta_j]_{i,j}$ si può riscrivere l'equazione di Eulero-Lagrange come:

$$C[P_i^{m+1}] = 0 \tag{1.27}$$

Inoltre utilizzando le matrici

$$U_e := \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} \text{ e } U_o := \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix}$$

si ha che si possono scomporre le componenti di $[P_i^{m+1}]$ come

$$[P_i^{m+1}] = U_e[P_{2i}^{m+1}] + U_o[P_{2i+1}^{m+1}]$$

Si definiscono, allora, $A := CU_o$ e $B = CU_e$ e si ha che l'equazione 1.27 diventa:

$$A[P_{2i+1}^{m+1}] = -B[P_i^m] \tag{1.28}$$

Si nota che A è sempre quadrata e simmetrica.

Esempio 1.18. `Schema Variazionale con Clothoid Discreto`

[11] Si definisce la curvatura nel punto $P_i^k = (x_i, y_i)$ come:

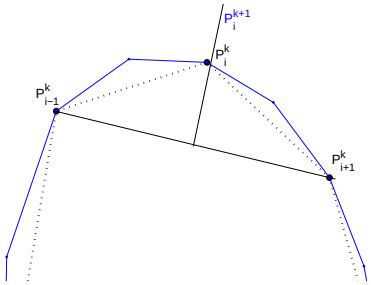
$$K_i = K(P_i) = 2\frac{(x_i - x_{i-1})(y_{i+1} - y_i) - (x_{i+2} - x_i)(y_i - y_{i-1})}{\|P_i - P_{i-1}\| \, \|P_i - P_{i+1}\| \, \|P_{i+1} - P_{i-1}\|}$$

Si vuole definire uno schema in cui le poligonali raffinate siano dei clothoid discreti. *La poligonale raffinata P^1 si dice* clothoid discreto *se soddisfa le seguenti proprietà:*

i) $\left\|P_{i-1}^1 - P_i^1\right\| = \left\|P_i^1 - P_{i+1}^1\right\|$ *per i nuovi punti calcolati con la legge, non per quelli pari.*

ii) $\Delta^2 K_i = K_{i-1} - 2k_i + k_{i+1} = 0$

Si ha che il nuovo punto P_{2i+1}^{k+1} dipende dai punti $P_{i-2}^k, ..., P_{i+2}^k$ del passo precedente; per *i)* P_i^{k+1} deve essere sulla perpendicolare alla retta che unisce P_{i-1}^k e P_{i+1}^k. Si richiede poi, per *ii)*, che il nuovo punto P_i^{k+1} soddisfi l'equazione non lineare $\Delta^2 K_i^{k+1} = 0$. Si risolve questo problema linearizzando l'equazione sostituendo P^k invece di P^{k+1} nel denominatore dell'espressione della curvatura. In questo modo si ottiene ogni P_i^{k+1}



risolvendo un sistema lineare 2x2 in cui si sostituisce P_i^k. Si ottiene, allora, la seguente legge:

$$\begin{cases} x_{2i}^{k+1} = \frac{-(\bar{y}-\bar{x})(x_i-x_{i+1})-y_i x_{i+1}+x_i y_{i+1}+\frac{2\|\bar{x}-x_i,\bar{y}-y_i\|^3}{4(k_i+k_{i+1})}}{y_{i+1}-y_i+m(x_i-x_{i+1})} \\ y_{2i}^{k+1} = m x_{2i}^{k+1} - m\bar{x} + \bar{y} \end{cases}$$

con $P_i^k = (x_i, y_i)$, $(\bar{x}, \bar{y})$ punto medio tra i punti P_i^k e P_{i+1}^k e $m = -\frac{x_{i+1}-x_i}{y_{i+1}-y_i}$ coefficiente angolare della retta perpendicolare alla retta passante per P_i^k e P_{i+1}^k.
Lo schema è uniforme per come sono definiti i nuovi punti ma non è stazionario perché ad ogni passo i coefficienti utilizzati cambiano.
Come ha dimostrato Kobbelt in [12], questo schema produce delle curve di alta qualità sfruttando le proprietà dell'approccio variazionale.

1.7. **Interpolazione Non Uniforme Stazionario**

In questa classe si studiano gli schemi non uniformi i cui punti, si ricorda, possono essere parametrizzati come $\{(t_i, P_i^k) : i \in \mathbb{Z}\}$. Gli schemi sono stazionari quindi, le leggi anche se dipendono dal posizione del nuovo punto, non cambiano da un livello all'altro.
Ci sono diversi esempi ma quello più significativo è un adattamento dello schema a quattro punti dell'esempio 1.11.

Esempio 1.19. `NULI`

In [18] Casciola, Beccari e Romani hanno studiato un particolare tipo di schema non uniforme, stazionario, lineare, interpolante che è un adattamento dello schema a quattro punti dell'esempio 1.11.
Data la poligonale iniziale $P^0 = \{P_0^0, ..., P_N^0\}$, si definisce una parametrizzazione, detta

centripeta, come:

$$x_i = x_{i-1} + \sqrt{\left\| P_i^0 - P_{i-1}^0 \right\|_2} \tag{1.29}$$

Per definire la partizione nodale $T = \{t_0, t_1, .., t_N\}$ associata alla poligonale iniziale si definiscono i parametri $\{\lambda_j^0\}$ con $\lambda_j \in [0,1]$, $j = 1, ..., N$. Allora T si definisce con la seguente legge:

$$t_{2i} := x_i^0 \text{ e } t_{2i+1} := x_i^0 + \lambda_i^0 (x_{i+1}^0 - x_i^0)$$

Si nota che, con questa definizione i nodi t_i non sono equispaziati quindi lo schema non è uniforme. Si definiscono, allora, anche gli intervalli nodali, che, si ricorda, hanno la seguente espressione:

$$d_i^0 := t_{i+1}^0 - t_i^0$$

Anche per questo schema si utilizza il metodo semi-regolare per la definizione dei nuovi nodi che saranno, quindi:

$$t_{2i}^{k+1} := t_i^k \text{ e } t_{2i+1} := \frac{t_i^k}{2}$$

quindi si hanno anche le seguenti definizioni degli intervalli nodali d_i e dei parametri λ_i

$$\begin{cases} \lambda_{2^k i-1}^k = \lambda_{i-1}^0 \\ \lambda_{2^k i}^k = \lambda_{2^k(i+1)-1}^k = \lambda_i^0 \end{cases} \quad \text{e } d_2^{k+1} i = d_{2i+1}^{k+1} = \frac{d_i^k}{2}$$

Per ogni lato della poligonale, inoltre, si definiscono i coefficienti c_0, c_1, c_2 e c_3 secondo la seguente legge:

$$\text{se } \lambda_i^k \in]0, \frac{1}{2}[\Rightarrow \begin{cases} c_0 = \frac{\lambda_i^k (d_i^k)^2}{8(\lambda_i^k - 1) d_{i-1}^k (d_{i-1}^k + d_i^k)} \\ c_1 = \frac{\lambda_i^k [-d_i^k d_{i+1}^k + d_i^k d_{i-1}^k + 4 d_{i+1}^k d_{i-1}^k - (d_i^k)^2] - 2 d_{i-1}^k (d_i^k + 2 d_{i+1}^k)}{8(\lambda_i^k - 1) d_{i-1}^k (d_{i-1}^k + d_i^k)} \\ c_2 = \frac{\lambda_i^k [3(d_i^k)^2 + 5 d_i^k d_{i+1}^k + 3 d_i^k d_{i-1}^k + 4 d_{i+1}^k d_{i-1}^k] - 2(d_{i-1}^k + d_i^k)(d_i^k + 2 d_{i+1}^k)}{8(\lambda_i^k - 1) d_{i-1}^k (d_{i-1}^k + d_i^k)} \\ c_3 = \frac{(2 - 3\lambda_i^k)(d_i^k)^2}{8(\lambda_i^k - 1) d_{i-1}^k (d_{i-1}^k + d_i^k)} \end{cases}$$

$$\text{se } \lambda_i^k \in]\frac{1}{2}, 1[\Rightarrow \begin{cases} c_0 = \frac{(1 - 3\lambda_i^k)(d_i^k)^2}{8 \lambda_i^k d_{i-1}^k (d_i^k + d_{i-1}^k)} \\ c_1 = \frac{(\lambda_i^k - 1)[3(d_i^k)^2 + 5 d_i^k d_{i-1}^k + 3 d_i^k d_{i+1}^k + 4 d_{i+1}^k d_{i-1}^k] + 2(d_{i-1}^k + d_i^k)(d_{i+1}^k + 2 d_{i-1}^k)}{8 \lambda_i^k d_{i-1}^k (d_{i+1}^k + d_i^k)} \\ c_2 = \frac{(\lambda_i^k - 1)[-d_i^k d_{i-1}^k + d_i^k d_{i+1}^k + 4 d_{i+1}^k d_{i-1}^k - (d_i^k)^2] + 2 d_{i+1}^k (d_i^k + 2 d_{i-1}^k)}{8 \lambda_i^k d_{i+1}^k (d_{i-1}^k + d_i^k)} \\ c_3 = \frac{(\lambda_i^k - 1)(d_i^k)^2}{8 \lambda_i^k d_{i+1}^k (d_{i+1}^k + d_i^k)} \end{cases}$$

allora si può dare la definizione dello shema come:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = c_0 P_{i-1}^k + c_1 P_i^k + c_2 P_{i+1}^k + c_3 P_{i+2}^k \end{cases} \tag{1.30}$$

Si osserva che, per come sono definiti i parametri λ ai passi successivi al primo, si ha che lo schema è stazionario.
Se i valori di d_i al passo iniziale sono uguali si ha che lo schema è uniforme, quindi bisogna utilizzare delle parametrizzazioni iniziali non uniformi. Inoltre se, oltre ad avere uno schema uniforme, si scelgono i parametri $\lambda_i^0 = \frac{1}{2} \forall i$ si ha che $c_0 = c3 = -\frac{1}{16}$ e $c_1 = c_2 = \frac{9}{16}$; cioè si ottine lo schema a quattro punti di Dubuc-Deslauries uniforme e stazionario dell'esempio 1.11.

1.8. **Interpolazione Non Uniforme Non Stazionario Lineare**

In questa classe si classificano tutti quegli schemi che non sono uniformi e non sono stazionari. Ciò significa che ad ogni passo di suddivisione i coefficienti della maschera cambiano e che la parametrizzazione risulta essere non uniforme.

Esempio 1.20. `Schema non uniforme e non stazionario`

Levin in [19] ha studiato un'esempio di schema in questa classe utilizzando un adattamento del metodo a quattro punti di Dubuc-Deslauries dell'esempio 1.14. Come maschera si utilizza:

$$a_i^k(x) = \frac{(x+1)^2}{2}\left(\frac{1}{x} - 2\omega_i^k x^{-3}(x-1)^2(x^2+1)\right)$$

Invece di utilizzare, come nell'esempio 1.14, ω, si hanno diversi valori per questo coefficiente. Nell'esempio 1.14 si aveva un unico ω; quando si cercavano i nuovi punti P_{2i+1} si utlizzava lo stesso valore. Invece in questo schema si hanno tanti ω quanti sono i punti della poligonale e quindi leggi diverse sia per ogni P_{2i+1} che per ogni livello. Lo schema risultante dalla maschera è:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = -\omega_i^k \left(P_i^k + P_{i+1}^k\right) + \left(\frac{1}{2} + \omega_i^k\right)\left(P_{i-1}^k + P_{i+2}^k\right) \end{cases} \tag{1.31}$$

Per i passi successivi al primo, come si vede in figura 1.20, si adotta la seguente legge:

$$\omega_{2i}^{k+1} = \omega_{2i+1}^{k+1} = \frac{\omega_i^k}{2}$$

Come per l'analisi dello schema a quattro punti uniforme si hanno delle restrizioni nella scelta dei coefficienti ω_i^0 iniziali. Per avere delle curve limite con poche irregolarità gli ω_i^0 dovrebbero essere scelti nell'intervallo $(0, \frac{1}{8})$.

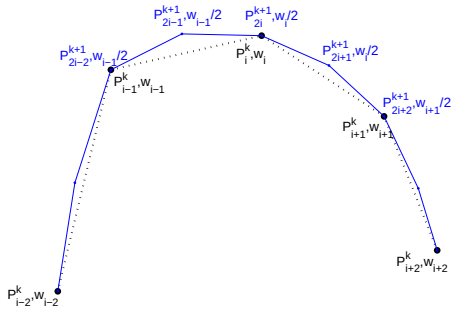


Figura 1.11: ω_i per il passo k+1

1.9. **Non Lineare**

In quest'ultima sezione si parlerà di uno schema di suddivisione interpolante non lineare. Finora gli schemi erano lineari e avevano delle proprietà particolari, come essere o meno stazionari, uniformi ed essere di interpolazione o di approssimazione. Si potrebbe fare una classificazione analoga per gli schemi non lineari ma in letteratura non sono stati sviluppati tutti gli esempi. Gli schemi non lineari sono caratterizzati dal fatto che i nuovi punti non sono una combinazione lineare dei punti ai passi precedenti ma sono generati con altri procedimenti.

Esempio 1.21. `Schema tangent-driven`

Lo schema di suddivisione interpolante non lineare presentato da L. Liang, B. Zou, H. Zhao e X. Qiu, in [20], ha la seguente legge:

$$\begin{cases} P^{k+1}_{2i} = P^k_i \\ P^{k+1}_{2i+1} = \bar{P}^k + d^k_i \end{cases} \tag{1.32}$$

con $\bar{P}^k$ punto medio tra P^k_i e P^k_{i+1} e

$$d^k_i = \omega \left(T^k_i \left| P^k_{i-1} P^k_i \right| - T^k_{i+1} \left| P^k_{i+1} P^k_{i+2} \right| \right)$$

che rappresenta il vettore spostamento corrispondente a $P^k_i P^k_{i-1}$, dove T^k_i è il vettore tangente unitario al vertice P^k_i e ω è il parametro di tensione.
Tornando all'esempio 1.14 dello schema a quattro punti con parametro, si nota che la legge:

$$P^{k+1}_{2i+1} = (P^k_i + P^k_{i+1}) \left(\omega + \frac{1}{2} \right) - \omega (P^k_{i-1} + P^k_{i+2})$$

si può riscrivere come:

$$P_{2i+1}^{k+1} = \frac{1}{2}(P_i^k + P_{i+1}^k) + \omega\left[(P_i^k - P_{i-1}^k) + (P_{i+1}^k - P_{i+2}^k)\right]$$

In quell'esempio il vettore spostamento è dato da:

$$d_i^k = \omega\left[(P_i^k - P_{i-1}^k) + (P_{i+1}^k - P_{i+2}^k)\right]$$

ed è una combinazione lineare dei punti al passo precedente.
In questo caso si vuole una combinzione non lineare e si introduce il vettore T_i^k che è dato, vedi [20], da

$$T_i^k := \frac{P_{i+1}^k - P_{i-1}^k}{\left\|P_{i+1}^k - P_{i-1}^k\right\|}$$

Quindi questo schema risulta essere non lineare perché il punto P_{2i+1}^{k+1} sarà una combinazione del punto medio tra P_i^k e P_{i+1}^k e del vettore spostamento che dipende dai vettori tangenti degli stessi punti e dalla distanza tra i punti P_{i-1}^k, P_i^k e $P_{i+1}^k P_{i+2}^k$.
Questo schema, inoltre, non è stazionario perché ad ogni passo cambiano i coefficienti.
Infine, siccome questo schema deriva direttamente dallo schema dell'esempio 1.14, il parametro di tensione ω, per avere delle curve accettabili, deve essere in $\left]0, \frac{1}{4}\right[$.

Capitolo 2

Applicazioni alle Immagini

L'idea degli schemi di suddivisione per curve (caso univariato) può essere estesa alle superfici (caso bivariato); le superfici di suddivisione furono introdotte nel campo della modellazione 3D nel 1978 con due articoli ad opera di Catmull e Clarck e di Doo e Sabin. Ora, in alternativa alle B-spline e alle NURBS, le superfici di suddivisione sono largamente utilizzate in diverse applicazioni, come la computer graphics, videogiochi 3D e film d'animazione. L'idea delle superfici di suddivisione è la stessa delle curve di suddivisione; cioè, partendo da un'insieme di vertici iniziali 3D, dopo un certo numero di raffinamenti in cui si aggiungono dei nuovi vertici ad ogni passo, si otterrà una superficie liscia e ben definita.
In questa trattazione non si vogliono approfondire le problematiche relative proprio alle superfici di suddivisione, ma si vuole focalizzare l'attenzione sugli schemi di suddivisione bivariati e alle loro applicazioni alle immagini (funzioni scalari bivariate). In particolare un'immagine digitale è rappresentata da una matrice bidimensionale, i cui elementi sono chiamati *pixel*. Esistono due tipologie fondamentali di immagini: le immagini rappresentate nella scala di grigi e le immagini a colori.
Le prime hanno come intervallo di rapprentazione $[0, 2^8 - 1] = [0, 255]$. Il valore 0 è il valore minimo di luminosità ed è associato al colore nero; invece il valore 255 rappresenta il massimo di luminosità ed è associato al colore bianco. I valori compresi nel range rappresentano tutte le tonalità del grigio.
Le immagini a colori, invece, hanno come intervallo di rappresentazione
$[0, 256 \times 256 \times 256 - 1]$, perché ogni set di 256 valori rappresenta le diverse sfumature di rosso, di blu e di verde. Combinando opportunamente la tonalità di rosso, di blu e di verde, si ottengono tutti colori. Il valore 0 rappresenta ancora il nero e in questo caso il bianco è associato al valore $256 \times 256 \times 256 - 1$. Per questo tipo di immagini non si crea una sola matrice, ma tre matrici differenti ognuna per ogni colore fondamentale. Quindi, se prima si avevano schemi di suddivisione applicati a poligonali iniziali, che potevano essere rappresentate da vettori di punti 2D; ora si vuole applicare il procedimento di suddivisione ad una matrice di punti 1D o 3D. Intuitivamente, con i processi di suddivi-

sione, ad ogni passo, si aggiungono nuovi elementi tra gli elementi della matrice al passo precedente. Si parte, cioè, dalla matrice al passo iniziale di dimensione NxM e, con un passo di suddivisione, si aggiungono esattamente $(N-1)$x$(M-1)$ nuovi elementi così la nuova matrice, al passo uno, avrà dimensione $2(N-1)$x$2(M-1)$. Il procedimento può essere iterato e si avranno matrici sempre più grandi. I dettagli relativi agli schemi di suddivisione per matrici sono riportati nella prima sezione di questo capitolo.
Una volta introdotti questi nuovi metodi, si spiegherà come applicarli alle immagini digitali ed in quali campi è possibile utilizzarli. In particolare la seconda sezione riguarda lo scaling di un'immagine e la terza sezione si occuperà di compressione.

2.1 Metodi di suddivisione bivariati

L'obiettivo di questa sezione è quello di introdurre metodi di suddivisione bivariati, il metodo generale differisce a seconda che lo schema univariato utilizzato sia di interpolazione o di approssimzione. Negli schemi bivariati si utilizzano combinazioni di schemi univariati; come si è appreso dalla teoria del primo capitolo si conoscono diversi schemi univariati, quindi si possono avere altrettanti schemi bivariati. In questa trattazione sono presenti le due principali categorie di classificazione di schemi di suddivisione, cioè gli schemi approssimanti e gli schemi interpolanti. Per ognuna di queste categorie esiste una diversa definizione di schemi bivariati. Si vuole, allora, dare un'idea di come procedere con gli schemi bivariati sia partendo da schemi approssimanti che da schemi interpolanti. Sono presenti anche degli esempi di schemi stazionari e uniformi, quali i metodi derivati da B-spline, gli schemi di interpolazione a N punti e uno schema non lineare.

2.1.1 Schemi di approssimazione

In questa sezione si vuole far vedere qual è lo schema bivariato derivante dagli schemi approssimanti, stazionari ed uniformi presentati negli esempi 1.1, 1.2 e 1.3.
Sia P^0 una matrice di dimensione N x M, i cui elementi sono indicati come $P(i,j)$ allora la nuova matrice P^1, al passo successivo, sarà di dimensione $2N$x$2M$. Il primo passo consiste nel definire una matrice di appoggio A di dimensione $2N$x$2M$ i cui elementi siano tutti nulli tranne alcuni, dati dalla lagge: $A(2i+1, 2j+1) = P^0(i,j)$ con $i = 0, ..., N$ e $j = 0, ..., M$. In pratica la matrice A è una riproduzione della matrice P^0. Simbolicamente:

$$P^0 = \begin{pmatrix} p^0(0,0) & p^0(0,1) & p^0(0,2) \\ p^0(1,0) & p^0(1,1) & p^0(1,2) \\ p^0(2,0) & p^0(2,1) & p^0(2,2) \end{pmatrix} \Rightarrow A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p^0(0,0) & 0 & p^0(0,1) & 0 & p^0(0,2) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p^0(1,0) & 0 & p^0(1,1) & 0 & p^0(1,2) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p^0(2,0) & 0 & p^0(2,1) & 0 & p^0(2,2) \end{pmatrix}$$

Si considera, ora, uno schema di suddivisione di approssimazione con maschera $a = (a_1, ..., a_h)$. Si considera tale maschera un vettore e si calcola il prodotto $a^T a$ ottenendo una matrice $\mathbf{M}$, anch'essa chiamata *maschera*, di dimensione hxh. Ora, per avere la nuova matrice al passo 1 si calcola il prodotto di convoluzione tra la matrice A e la matrice maschera $\mathbf{M}$. Per i passi successivi si applica esattamente lo stesso procedimento. Tale schema può essere definito da schemi uniformi e stazionari, in particolare vengono forniti gli esempi e tutti i dettagli degli schemi degli esempi 1.1, 1.2 e 1.3.

Esempio 2.1. `B-spline lineare`

Il metodo B-spline lineare univariato, dell'esempio 1.1, è dato dal seguente schema:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{1}{2}\left(P_i^k + P_{i+1}^k\right) \end{cases}$$

ed ha maschera $a = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$ La matrice maschera è, quindi, data da:

$$\mathbf{M} = \begin{pmatrix} \frac{1}{2} \\ 1 \\ \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix} \tag{2.1}$$

In questo caso la convoluzione, applicata ad una matrice $2N$x$2M$, definisce una matrice di dimensione $2N-2$x$2M-2$, allora si definisce la matrice P^k aggiungendo due righe e due colonne nulle, nel modo seguente:

$$P^k = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

dove $\bullet$ sono gli elementi calcolati con la convoluzione.

Esempio 2.2. `B-spline quadratica`

Per lo schema quadratico, dell'esempio 1.2 si procede esattamente allo stesso modo dello schema lineare. Si ricorda che la B-spline quadratica ha come legge:

$$\begin{cases} P_{2i+1}^{k+1} = \frac{1}{4}\left(P_i^k + 3P_{i+1}^k\right) \\ P_{2i}^{k+1} = \frac{1}{4}\left(3P_i^k + P_{i+1}^k\right) \end{cases}$$

e, come maschera

$$a = \left(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}\right)$$

Allora si calcola la matrice $\mathbf{M}$ come:

$$\mathbf{M} = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \\ \frac{3}{4} \\ \frac{1}{4} \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{pmatrix} \tag{2.2}$$

La convoluzione, applicata ad una matrice $2N$x$2M$, con questa particolare maschera 4x4, produce una matrice $2N-4$x$2N-4$, allora si aggiungono due righe e due colonne nulle per avere la matrice P^{k+1} cercata. Simbolicamente:

$$P^k = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

dove $\bullet$ sono gli elementi calcolati con la convoluzione.

Esempio 2.3. `B-spline cubica`

Infine, si ricorda che lo schema univariato B-spline cubico, dell'esempio 1.3, è dato dalla legge

$$\begin{cases} P_{2i+1}^{k+1} = \frac{1}{8}\left(P_{i-1}^k + 6P_i^k + P_{i+1}^k\right) \\ P_{2i}^{k+1} = \frac{1}{8}\left(4P_i^k + 4P_{i+1}^k\right) \end{cases}$$

con maschera $a = \left(\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8}\right)$. Quindi la matrice $\mathbf{M}$ sarà:

$$\mathbf{M} = \begin{pmatrix} \frac{1}{8} \\ \frac{1}{2} \\ \frac{3}{4} \\ \frac{1}{2} \\ \frac{1}{8} \end{pmatrix} \begin{pmatrix} \frac{1}{8} & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} & \frac{1}{8} \end{pmatrix} = \frac{1}{64} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} \tag{2.3}$$

In questo esempio si ha che la convoluzione ha come risultato una matrice di dimensione $2N-6$x$2N-6$ allora, in analogia con gli altri esempi, si aggiungono tre righe e tre

colonne nulle per avere l'espressione della matrice P^{k+1}. Simbolicamente:

$$P^k = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \bullet\cdots\bullet & 0 & 0 & 0 \\ & \vdots & & \vdots\cdots\vdots & & \vdots & \\ 0 & 0 & 0 & \bullet\cdots\bullet & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

dove $\bullet$ sono gli elementi calcolati con la convoluzione.

2.1.2 Schemi di interpolazione

In questa sezione si affronteranno le tematiche relative alla definizione di schemi bivariati derivati da metodi interpolanti, in particolare, di schemi uniformi e stazionari. Verrà esposto il metodo generale per definire lo schema bivariato, si parlerà di due tecniche per determinare dei punti particolari e, infine, verranno forniti due esempi di tipologie di schemi, lo schema a N punti, dell'esempio 1.13 e lo schema non lineare tangent-driven, esempio 1.21.
Ora, sia P^0 una matrice di dimensione N x M, i cui elementi sono indicati come $P(i,j)$ allora la nuova matrice P^1, al passo successivo, sarà di dimensione $(2N-1)$x$(2M-1)$. Al passo k-esimo la matrice P^k avrà dimensione $\left[2^k(N-1)+1\right]$x$\left[2^k(M-1)+1\right]$. Si consideri allora la matrice P^k di cui si vuole trovare il raffinamento P^{k+1}. La matrice P^{k+1} ha, nelle posizioni con entrambi gli indici pari, l'esatta riproduzione degli elementi della matrice P^k, cioè

$$P^{k+1}(2i,2j) = P^k(i,j)$$

$$P^k = \begin{pmatrix} p^k(0,0) & p^k(0,1) & p^k(0,2) \\ p^k(1,0) & p^k(1,1) & p^k(1,2) \\ p^k(2,0) & p^k(2,1) & p^k(2,2) \end{pmatrix} \Rightarrow P^{k+1} = \begin{pmatrix} p^k(0,0) & \circ & p^k(0,1) & \circ & p^k(0,2) \\ \circ & \circ & \circ & \circ & \circ \\ p^k(1,0) & \circ & p^k(1,1) & \circ & p^k(1,2) \\ \circ & \circ & \circ & \circ & \circ \\ p^k(2,0) & \circ & p^k(2,1) & \circ & p^k(2,2) \end{pmatrix}$$

Nelle righe pari, cioè dove sono già presenti dei valori, si applica la seguente legge:

$$P^{k+1}(2i+1,2j) = \sum_l \gamma_l P^k(i-l,j) \tag{2.4}$$

dove γ_l rappresenta i coefficienti del metodo di suddivisione univariato scelto. Ciò significa che per il calcolo del nuovo punto si applica uno schema di suddivisione univari-

ato al vettore rappresentato dalla riga in questione.

$$P^k = \begin{pmatrix} \mathbf{p^k(0,0)} & \mathbf{p^k(0,1)} & \mathbf{p^k(0,2)} \\ p^k(1,0) & p^k(1,1) & p^k(1,2) \\ p^k(2,0) & p^k(2,1) & p^k(2,2) \end{pmatrix} \Rightarrow$$

$$P^{k+1} = \begin{pmatrix} p^k(0,0) & \mathbf{p^{k+1}(0,1)} & p^k(0,1) & \mathbf{p^{k+1}(0,3)} & p^k(0,2) \\ \circ & \circ & \circ & \circ & \circ \\ p^k(1,0) & p^{k+1}(1,1) & p^k(1,1) & p^{k+1}(1,3) & p^k(1,2) \\ \circ & \circ & \circ & \circ & \circ \\ p^k(2,0) & p^{k+1}(2,1) & p^k(2,1) & p^{k+1}(2,3) & p^k(2,2) \end{pmatrix} \quad (2.5)$$

In cui si ha che gli elementi della matrice P^{k+1} evidenziati in grasseto sono il risultato di uno schema di suddivisione, che ha come coefficienti γ_l, applicato alla riga evidenziata in grassetto della matrice P^k. Si procede, poi, allo stesso modo, con le colonne in posizione pari, utilizzando la legge:

$$P^{k+1}(2i, 2j+1) = \sum_l \gamma_l P^k(i, j-l) \quad (2.6)$$

anche in questo caso γ_l rappresenta lo schema di suddivisione; tale metodo verrà applicato alle colonne della matrice P^k ottenendo gli elementi nella nuova matrice P^{k+1}.

$$P^k = \begin{pmatrix} \mathbf{p^k(0,0)} & p^k(0,1) & p^k(0,2) \\ \mathbf{p^k(1,0)} & p^k(1,1) & p^k(1,2) \\ \mathbf{p^k(2,0)} & p^k(2,1) & p^k(2,2) \end{pmatrix} \Rightarrow$$

$$P^{k+1} = \begin{pmatrix} p^k(0,0) & p^{k+1}(0,1) & p^k(0,1) & p^{k+1}(0,3) & p^k(0,2) \\ \mathbf{p^{k+1}(1,0)} & \circ & p^{k+1}(1,2) & \circ & p^{k+1}(1,2) \\ p^k(1,0) & p^{k+1}(1,1) & p^k(1,1) & p^{k+1}(1,3) & p^k(1,2) \\ \mathbf{p^{k+1}(3,0)} & \circ & p^{k+1}(3,2) & \circ & p^{k+1}(3,2) \\ p^k(2,0) & p^{k+1}(2,1) & p^k(2,1) & p^{k+1}(2,3) & p^k(2,2) \end{pmatrix} \quad (2.7)$$

Da questo piccolo esempio si vede che i risultati evidenziati in grassetto nella matrice P^{k+1} sono dati dagli elementi della colonna evidenziata nella matrice P^k. Infine, per i rimanaenti punti si utilizza la seguente legge:

$$P^{k+1}(2i+1, 2j+1) = \sum_{l,\nu} \beta_{l,\nu} P^k(i-l, j-\nu) \quad (2.8)$$

e si ottiene l'espressione finale della matrice P^k

$$P^{k+1} = \begin{pmatrix} p^k(0,0) & p^{k+1}(0,1) & p^k(0,1) & p^{k+1}(0,3) & p^k(0,2) \\ p^{k+1}(1,0) & \mathbf{p^{k+1}(1,1)} & p^{k+1}(1,2) & \mathbf{p^{k+1}(1,3)} & p^{k+1}(1,2) \\ p^k(1,0) & p^{k+1}(1,1) & p^k(1,1) & p^{k+1}(1,3) & p^k(1,2) \\ p^{k+1}(3,0) & \mathbf{p^{k+1}(3,1)} & p^{k+1}(3,2) & \mathbf{p^{k+1}(3,3)} & p^{k+1}(3,2) \\ p^k(2,0) & p^{k+1}(2,1) & p^k(2,1) & p^{k+1}(2,3) & p^k(2,2) \end{pmatrix}$$

$\beta_{l,\nu}$ rappresenta i coefficienti dello schema bivariato vero e proprio. Come anticipato, in questa trattazione sono studiati due diversi metodi per determinare questi coefficienti. Il primo è il prodotto tensoriale tra due schemi univariati, utilizzato per gli schemi univariati a N punti [8]; e il secondo è il metodo presentato da L. Liang, B. Zou, H. Zhao e X. Qiu, in [20], utilizzato per lo schema non lineare tangent-driven.
Quindi, ricapitolando, si hanno le leggi:

$$\begin{cases} P^{k+1}(2i,2j) = P^k(i,j) \\ P^{k+1}(2i+1,2j) = \sum_l \gamma_l P^k(i-l,j) \\ P^{k+1}(2i,2j+1) = \sum_l \gamma_l P^k(i,j-l) \\ P^{k+1}(2i+1,2j+1) = \sum_{l,\nu} \beta_{l,\nu} P^k(i-l,j-\nu) \end{cases} \tag{2.9}$$

Di seguito sono riportati in dettaglio gli schemi bivariati determinati a partire da alcuni schemi noti del primo capitolo.

2.1. **Schema a N punti**

Per questo tipo di schemi si utilizza il prodotto tensoriare per determinare i valori dei coefficienti $\beta_{l,\nu}$. Sono riportati sue semplici schemi, cioè il due e il quattro punti.

Esempio 2.4. `Schema a due punti`

Il metodo univariato, dell'esempio 1.8, è dato dal seguente schema:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{1}{2}\left(P_i^k + P_{i+1}^k\right) \end{cases}$$

Il nuovo metodo bivariato sarà dato da:

$$\begin{cases} P^{k+1}(2i,2j) = P^k(i,j) \\ P^{k+1}(2i+1,2j) = \frac{1}{2}\left(P^k(i,j) + P^k(i+1,j)\right) \\ P^{k+1}(2i,2j+1) = \frac{1}{2}\left(P^k(i,j) + P^k(i,j+1)\right) \\ P^{k+1}(2i+1,2j+1) = \frac{1}{4}\left(P^k(i,j) + P^k(i,j+1) + P^k(i+1,j) + P^k(i+1,j+1)\right) \end{cases} \tag{2.10}$$

Quindi, in questo caso, per trovare il nuovo elemento in posizione $(2i+1, 2j+1)$ si ha bisogno di 4 elementi della matrice al passo precedente. Osservando graficamente si ha che:

$$P^k = \begin{pmatrix} \bullet & \bullet & \otimes \\ \bullet & \bullet & \otimes \\ \cdot & * & * \end{pmatrix} \Rightarrow P^{k+1} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \bullet & \cdot & & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & * & \cdot \end{pmatrix}$$

in cui si ha che i simboli $*$ in P^k rappresentano gli elementi delle righe a cui si applica lo schema per avere il corrispondente elemento $*$ in P^{k+1}; $\otimes$ in P^k rappresentano gli elementi

delle colonne necessari per avere il corrispondente elemento $\otimes$ in P^{k+1} e i simboli $\bullet$ in P^k sono gli elementi convolti nel prodotto tensoriale che permettono di calcolare $\bullet$ in P^{k+1}.

Esempio 2.5. `Schema a quattro punti`

Si vuole ora osservare il metodo bivariato per lo schema a quattro punti. Dall'esempio 1.11 si ha che il metodo a quattro punti univariato è dato dalla seguente legge:

$$\begin{cases} P_{2i}^{k+1} = P_i^k \\ P_{2i+1}^{k+1} = \frac{9}{16}\left(P_i^k + P_{i+1}^k\right) - \frac{1}{16}\left(P_{i-1}^k + P_{i+2}^k\right) \end{cases}$$

il cui schema bivariato associato è dato da:

$$\begin{cases} P^{k+1}(2i,2j) = P^k(i,j) \\ P^{k+1}(2i+1,2j) = \frac{9}{16}\left(P^k(i,j) + P^k(i+1,j)\right) - \frac{1}{16}\left(P^k(i-1,j) + P^k(i+2,j)\right) \\ P^{k+1}(2i,2j+1) = \frac{9}{16}\left(P^k(i,j) + P^k(i,j+1)\right) - \frac{1}{16}\left(P^k(i,j-1) + P^k(i,j+2)\right) \\ \begin{aligned} P^{k+1}(2i+1,2j+1) =& \frac{1}{16^2}P^k(i-1,j-1) - \frac{9}{16^2}P^k(i-1,j) \\ & - \frac{9}{16^2}P^k(i-1,j+1) + \frac{81}{16^2}P^k(i-1,j+2) - \frac{9}{16^2}P^k(i,j-1) + \\ & \frac{81}{16^2}P^k(i,j) + \frac{81}{16^2}P^k(i,j+1) - \frac{9}{16^2}P^k(i,j+2) \\ & - \frac{9}{16^2}P^k(i+1,j-1) + \frac{81}{16^2}P^k(i+1,j) + \frac{81}{16^2}P^k(i+1,j+1) \\ & - \frac{9}{16^2}P^k(i+1,j+2) + \frac{1}{16^2}P^k(i+2,j-1) - \frac{9}{16^2}P^k(i+2,j) \\ & - \frac{9}{16^2}P^k(i+2,j+1) + \frac{1}{16^2}P^k(i+2,j+2) \end{aligned} \end{cases} \tag{2.11}$$

Graficamente:

$$P^k = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \otimes \\ \bullet & \bullet & \bullet & \bullet & \otimes \\ \bullet & \bullet & \bullet & \bullet & \otimes \\ \bullet & \bullet & \bullet & \bullet & \otimes \\ * & * & * & * & \cdot \end{pmatrix} \Rightarrow P^{k+1} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \bullet & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

anche in questo caso si ha che i simboli $*$ in P^k rappresentano gli elementi delle righe a cui si applica lo schema per avere il corrispondente elemento $*$ in P^{k+1}; $\otimes$ in P^k

rappresentano gli elementi delle colonne necessari per avere il corrispondente elemento $\otimes$ in P^{k+1} e i simboli $\bullet$ in P^k sono gli elementi convolti nel prodotto tensoriale che permettono di calcolare $\bullet$ in P^{k+1}. Per il prodotto tensoriale dello schema a quattro punti sono necessari 16 elementi.

Si ha, inoltre, che si possono definire i metodi bivariati per un numero di punti n più grande in cui, per ogni elemento con indice $(2i+1, 2j+1)$, quelli calcolati con il prodotto tensoriale, ci sarà bisogno di n^2 elementi della matrice al passo precedente.

2.2. **Schema non lineare**

L. Liang, B. Zou, H. Zhao e X. Qiu, [20], hanno studiato lo schema non lineare del tangent-driven, esempio 1.21, ed hanno anche fornito un modo per definire lo stesso schema bivariato. Questa idea si può applicare a qualunque schema a n punti di interpolazione e serve a determinare i coefficienti $\beta_{l,\nu}$ dell'equazione 2.6.
In uno schema univariato a n punti, in generale, si applica un procedimento, G, che dati n punti consecutivi calcola il nuovo punto al passo successivo, cioè

$$P^{k+1}_{2i+1} = G(P^k_j, P^k_{j+1}, ..., P^k_{j+n-1}) \tag{2.12}$$

con diverse interpretazioni di j che dipendono dallo schema scelto (schema a 4 punti $j = i-1, i, i+1, i+2$, schema a sei punti $j = i-2, i-1, i, i+1, i+2, i+3$ etc.).
Sia, ora, P^k la matrice a cui si deve applicare lo schema di suddivisione bivariato. La matrice, P^{k+1}, al passo successivo avrà, come elementi $P^{k+1}(2i, 2j)$, la replica degli elementi della matrice P^k; sulle righe e sulle colonne pari si applica lo schema univariato e, per gli elementi $P^{k+1}(2i+1, 2j+1)$ si applica lo schema univariato ad esattamente n elementi nella diagonale della matrice P^k relativa alla posizione degli indici (i, j). In formule:

$$\begin{cases} P^{k+1}(2i, 2j) = P^k(i, j) \\ P^{k+1}(2i+1, 2j) = G\left(P^k_(i, j), P^k_(i+1, j), ..., P^k(i+n-1, j)\right) \\ P^{k+1}(2i, 2j+1) = G\left(P^k_(i, j), P^k_(i, j+1), ..., P^k(i, j+n-1)\right) \\ P^{k+1}(2i+1, 2j+1) = G\left(P^k_(i, j), P^k_(i+1, j+1), ..., P^k(i+n-1, j+n-1)\right) \end{cases}$$

Queste sono le formule generali; di seguito è fornito un esempio di schema a quattro punti con tutti i dettagli.

Esempio 2.6. `Schema a quattro punti`

Sia ancora G il procedimento che, dati quattro punti, calcola il risultato di uno schema di suddivisione. Questa funzione G non è altro che il procedimento di suddivisione

applicato ai punti dispari (i punti pari sono replicati). Allora la definizione dello schema bivariato è la seguente:

$$\begin{cases} P^{k+1}(2i,2j) = P^k(i,j) \\ P^{k+1}(2i+1,2j) = G\left(P^k(i-1,j), P^k(i,j), P^k(i+1,j), P^k(i+2,j)\right) \\ P^{k+1}(2i,2j+1) = G\left(P^k(i,j-1), P^k(i,j), P^k(i,j+1), P^k(i,j+2)\right) \\ P^{k+1}(2i,2j+1) = G\left(P^k(i-1,j-1), P^k(i,j), P^k(i+1,j+1), P^k(i+2,j+2)\right) \end{cases} \tag{2.13}$$

Quindi, per i nuovi punti sulle righe e sulle colonne pari si utilizza il metodo non lineare univariato; invece, per gli elementi rimanenti si applica il metodo univariato a quattro punti della matrice P^k che sono sulla diagonale del punto corrispondente. Graficamente:

$$P^k = \begin{pmatrix} \bullet & \cdot & \cdot & \cdot & \otimes \\ \cdot & \bullet & \cdot & \cdot & \otimes \\ \cdot & \cdot & \bullet & \cdot & \otimes \\ \cdot & \cdot & \cdot & \bullet & \otimes \\ * & * & * & * & \cdot \end{pmatrix} \Rightarrow P^{k+1} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \bullet & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

quindi gli elementi $\bullet$ della matrice P^k permettono di calcolare il nuovo elemento $\bullet$ della matrice P^{k+1}.

Nel caso delle schema tangent-driven non bisogna far altro che sostituire il procedimento G con la legge di suddivisione dell'equazione 1.32.

2.2 Scaling di Immagini

Quando si lavora con immagini digitali si ha, spesso, la necessità di scalarle, cioè si ha bisogno di cambiare le dimensioni dell'immagine ma cercando di non perdere la qualità e i dettagli; questa operazione viene chiamata *scaling*. In letteratura esistono molti metodi di scaling di immagini ma si vuole proporre un metodo basato sugli schemi di suddivisione bivariati. Come anticipato nell'introduzione al capitolo, si ha che un'immagine digitale è rappresentata da una matrice I, di dimensione NxM, i cui elementi sono chiamati pixel. Sia k il fattore per il quale si sceglie di scalare l'immagine, allora l'immagine finale avrà dimensione kNxkM. Il problema è la determinazione del valore da attribuire ad ogni pixel della nuova immagine, chiamata I_s.
Si considera, come primo passo, ogni pixel dell'immagine finale I_s e gli si applica la trasformazione inversa per portarlo nello spazio dell'immagine originale I, si determinano i pixel di I più vicini al pixel considerato; poi si procede con un metodo di *filtering*. I metodi più conosciuti, in letteratura, sono il *Nearest Neighbor*, il *Bilinear Interpolation* e il *Bicubic Interpolation*.
Nel *Nearest Neighbor* il valore del pixel nell'immagine finale è posto uguale al valore del pixel più vicino nell'immagine originale.
Nel *Bilinar Interpolation* il valore del pixel nell'immagine I_s è ricavato interpolando linearmente prima lungo le righe e poi lungo le colonne (o viceversa) formate dai 4 pixel più vicini nell'immagine originale.
Nel *Bicubic Interpolation*, infine, il valore del pixel nell'immagine finale è calcolato interpolando con polinomi cubici prima lungo le righe e poi lungo le colonne (o viceversa) formate dai 16 pixel più vicini nell'immagine I.
Come anticipato prima, per ingrandire un'immagine, in particolare per raddoppiare, quadruplicare,... le dimensioni, si possono utilizzare i metodi di suddivisione bivariati. L'idea è molto semplice. Si ha una matrice che rappresenta l'immagine iniziale da ingrandire, allora basta applicare un metodo bivariato visto nella prima sezione ed avere una matrice di dimensioni doppie, che rappresenta l'immagine finale. Per quadruplicare le dimensioni bisogna applicare il metodo di suddivisione per due passi, per ingrandire otto volte bisogna applicare il metodo per tre passi, e così via.
I metodi di suddivvisione univariati sono successivi raffinamenti della poligonale iniziale; i metodi bivariati sono successivi raffinamenti della matrice iniziale, cioè dell'immagine iniziale. Se ad un'immagine, infatti, viene applicato uno schema di suddivisione bivariato si ha che l'immagine risultante ha dimensioni doppie, nel caso di schemi approssimanti, o di $(2N-1)$x$(2M-1)$, nel caso di schemi interpolanti, ed ha maggiori dettagli perché si aggiungono degli elementi.
Si possono utilizzare tutti i metodi visti ed ottenere risultati più o meno di qualità.

Osservazione 5. Nel caso $k = 2h$ $h \in \mathbb{N}$, gli schemi visti negli esempi 2.1 e 2.3 riproducono, rispettivamente, gli schemi Bilinear e Bicubic se l'interpolazione lineare è data

dal punto medio e il bicubico utilizza, come polinomi cubici di interpolazione, le funzioni B-spline cubiche.

2.3 Compressione di Immagini

La *compressione* di un segnale o di un'immagine digitale consiste nel ridurre la quantità di dati richiesti per rappresentare quel segnale o quell'immagine. Un set di dati viene compresso perché presenta delle informazioni ridondanti quindi si vuole risparmiare spazio durante la memorizzazione o ridurre il tempo di trasmissione.
In letteratura esistono diversi metodi, chiamati *codifiche* per la riduzione della ridondanza quali: la *Run-Length*, utilizzata quando i dai presentano ripetizioni consecutive di alcuni simboli; la *Codifica di Huffman*, che rappresenta simboli a probabilità di occorrenza decrescente mediante una codifica a lunghezza di codice crescente; la *DPCM (delta)*, che rappresenta i simboli originali come differenza tra il simbolo corrente e una predizione dei simboli precedenti; la *Codifica LZW* che è una tecnica massicciamente utilizzata nella compressione dati, utilizza una tabella per rappresentare sequenze ricorrenti di simboli e la *Codifica aritmetica* che rappresenta gruppi di simboli, in base alla probabilità di ricorrenza degli stessi, mediante una rappresentazione in floating point.
Questi metodi possono essere utilizzati per la compressione di segnali e, sono l'idea di base della compressione di immagini digitali. In questa sezione si spiegherà un nuovo motodo di compressione di segnali basata sugli schemi di suddivisione che poi sarà l'idea di base di un nuovo metodo di compressione di immagini.

2.3.1 Compressione di Segnali

Come appena anticipato, in questa sezione si vuole introdurre un nuovo metodo di compressione di segnali basata sugli schemi di suddivisioni visti nel primo capitolo. L'idea è di Floater e Dahlen, [20], e si può riassumere semplicemente. Dato un segnale campionato, si scelgono un piccolo numero di punti e si applica uno schema di suddivisione interpolante. I nuovi punti trovati saranno un'approssimazione del segnale iniziale. Si calcola, quindi, la differenza tra i nuovi dati ed i corrispondenti dati del segnale iniziale; si memorizzano, poi, solo le differenze maggiori di una certa tolleranza fissata. Si ripete la suddivisione e si memorizzano gli errori fino ad avere tutto il segnale iniziale. Per decomprimere il segnale basta applicare il metodo di suddivisione per i punti iniziali e, infine, aggiungere l'errore calcolato. Quindi la compressione si sintetizza in tre passaggi fondamentali: *Decomposizione*, *Compressione* e *Decompressione*.

Decomposizione
Siano $P_i \in \mathbb{R}$ $i = 1, ..., N$ i punti di un segnale campionato $f : \mathbb{Z} \to \mathbb{R}$ tale che $f(i) = P_i$. L'obiettivo è trovare una funzione approssimante f, $\hat{f} : \mathbb{Z} \to \mathbb{R}$, $\hat{f}$, che permetta di

rappresentare f con il minor numero di dati possibili. Scelto poi $M \in \mathbb{N}$ siano:

$$\Omega_k := \left\{2^{M-k} i \text{ t. c. } i \in \mathbb{Z}\right\} \text{ e } H_k := \left\{2^{M-k}(i + \frac{1}{2}) \text{ t. c. } i \in \mathbb{Z}\right\}$$

Con queste definizioni si può riscrivere la funzione discreta f come somma di funzioni discrete f_i

$$f = f_0 + f_1 + \cdots + f_M$$

dove f_0 è la proiezione di f nello spazio Ω_0, f_1 la proiezione di f nello spazio Ω_1, $\cdots$, f_N la proiezione dello spazio Ω_N.
Se $g : \Omega_k \to \mathbb{R}$ è un vettore di punti, si definisce $R(g) : H_r \to \mathbb{R}$ la legge di suddivisione interpolante che, dato un vettore di punti, calcola il risultato del procedimento di suddivisione. Se g è N-dimensionale $R(g)$ è un vettore di dimensione $2N - 1$.
Utilizzando tale procedimento R si calcolano i vari passi partendo da f_0 che rappresenta la poligonale iniziale.

$$\begin{gathered}
\hat{f}_1 = R(f_0) \\
\hat{f}_2 = R(\hat{f}_1) = R(f_1 + f_0) \\
\hat{f}_3 = R(\hat{f}_2) = R(f_2 + f_1 + f_0) \\
\cdots \\
\hat{f_M} = R(\hat{f_{M-1}}) = R(f_{M-1} + \cdots + f_1 + f_0)
\end{gathered}$$

In pratica f_i sono le funzioni esatte mentre $\hat{f}_i$ rappresentano le approssimazoni delle f_i ottenute utilizzando un processo di suddivisione.
Ora si definiscono le differenze al passo k-esimo come:

$$e_k : H_{k-1} \to \mathbb{R} \text{ con } e_k := f_k - R(f_0 + f_1 + ... + f_{k-1})$$

f_0 è la rappresentazione esatta della proiezione di f in Ω_0, mentre $e_1 = f_1 - R(f_1 + f_0) = f_1 - \hat{f}_1$ cioè rappresenta la differenza tra la funzione esatta f_1 e la sua approssimazione calcolata con il processo di suddivisione. Quindi $e_k = f_k - \hat{f}_k \ \forall k$
Si ottiene, in questo modo, la decomposizione $f_0, e_1, ..., e_N$.

Compressione
Una volta operata la decomposizione del segnale in $f_0, e_1, ..., e_N$ e fissata una tolleranza $\epsilon > 0$, si cercano le approssimazioni $\hat{f}_0, \hat{e_1}, ..., \hat{e_N}$ che hanno il numero minore possibile di elementi diversi da zero. Quindi si definisce:

$$\hat{e_k} := \begin{cases} e_k(x), \text{ se } |e_k(x)| > \epsilon \\ 0, \text{ se } |e_k(x)| \leq \epsilon \end{cases} \qquad \text{per} x \in K_{r-1}$$

con $k = 1, ..., N$. La compressione del segnale, allora, consiste nel salvare non tutti i valori degli errori e_k ma solo gli errori significativi rappresentati da $\hat{e_k}$. Quindi, invece

di salvare tutto il segnale, si salva solo cioò che rappresenta la poligonale iniziale, cioè f_0, e gli errori significativi $\hat{e_k}$. Semplicemente, quindi, invece di salvare f si salvano $f_0, \hat{e_1}, \cdots, \hat{e_N}$

Decompressione
Come appena spiegato, invece del segnale f si ha la sua compressione rappresentata da $f_0, \hat{e_1}, \cdots, \hat{e_N}$. Per decomprimere il segnale si utilizza f_0 come dato iniziale e si procede con il metodo di suddivisione R, ottenendo le funzioni $\hat{f}_r : H_{k-1} \to \mathbb{R}$.
Il valore del segnale approssimato sarà, allora:

$$\hat{f}_k := R(\hat{f}_0 + \hat{f}_1 + ... + \hat{f}_{k-1}) + \hat{e_k}$$

Quindi, partendo da f_0 si ricompone il segnale utilizzando il metodo di suddivisione. Il risultato della decompressione non sarà la funzione esatta f ma una sua approssimazione, l'errore commesso nella ricostruzione è dell'ordine della tolleranza fissata prima della compressione.

Il vantaggio di questo metodo consiste in un risparmio di spazio durante la memorizzazione. Infatti, se prima bisognava inviare un segnale molto lungo, ora si memorizzano gli errori in un vettore e si inviano solamente i punti scelti e questo vettore. L'obiettivo di questo procedimento è fare in modo che il vettore delle differenze abbia meno valori possibili; a questo proposito si possono confrontare schemi diversi e si considera migliore quello che fornisce il vettore degli errori con il numero di zeri più grande.

2.3.2 Compressione di Immagini

La compressione di immagini digitali è un campo di ricerca in continua espansione, si sfrutta la caratteristica comune nella maggior parte delle immagini per cui pixel adiacenti sono correlati e in quanto tali contengono informazione ridondante. Si cerca quindi una rappresentazione *meno correlata* delle immagini, eliminando le *ripetizioni*. Vengono, quindi, scartate informazioni *irrilevanti* per il tipo di fruizione richiesta: sfruttando le caratteristiche del sistema visivo (HVS, Human Visual System), si omette o si rappresenta in modo meno accurato quella parte del segnale per cui il ricevitore risulta meno *sensibile*.
Esistono due tipologie di compressione quali la *Compressione senza e con perdita (Lossless vs. Lossy compression)*. Nel caso lossless, l'immagine ricostruita dopo la compressione è numericamente identica all'originale. Invece, negli approcci di tipo lossy, viene introdotto un degrado rispetto all'originale;
Con la compressione lossless, cioè, non si perdono dati; invece, con la compressione lossy si perdono dati *meno importanti*. Ad esempio l'immagine ricostruita decomprimendo il file inganna l'occhio, ma contiene notevoli differenze. Solitamente tali differenze non risultano percettibili, in quanto la parte di informazione persa è comunque quella che

l'utente non avrebbe notato.
Un esempio di compressione lossless è l'algoritmo RLE (run lenght encoding, codifica della lunghezza delle sequenza) la cui idea di base è molto semplice. Infatti quando si trova una serie di pixel dello stesso colore, si codifica solo la lunghezza della serie e il colore. Per immagini geometriche, funziona molto bene. Se l'immagine ha ampie aree di colore uguale, si ottengono grossi risparmi; se, viceversa, ogni pixel ha un colore diverso, utilizzare questo tipo di compressione può peggiorare i risultati, cioè può occupare più memoria di quanta ne occupa l'immagine origiale.
Un esempio di compressione lossy è l'algoritmo JPEG che funziona particolarmente su immagini con molte sfumature (fotografie, incarnati, paesaggi). Lo svantaggio è che quando l'immagine ha un forte contrasto o passaggi bruschi di colore, si possono notare dei difetti (detti *artefatti*). L'algoritmo si può riassumere in tre passi fondamentali: l'immagine viene dapprima trasportata in un altro spazio colore, cioè si cambia il modello colore da RGB (o altro) a YUV.
Le immagini, infatti, sono solitamente rappresentate mediante tre componenti di colore (Red, Green, Blue - RGB) ma, per applicazioni di compressione viene utilizzata la rappresentazione YUV (luminanza + 2 componenti di crominanza). In queste applicazioni si usa questa rappresentazione perché l'occhio umano è più sensibile alla luminosità che al particolare colore, quindi si procede con il ridurre il livello di crominanza, mentre si lascia invariata la luminanza.
Come secondo passo, si divide l'immagine in diversi blocchi e, ciascuno di essi, viene trasformato usando la trasformata discreta del coseno, DTC. Con questi primi due passi si riduce la risoluzione della crominanza, si approssimano i colori e si eliminano i dettagli troppo fini per essere visti dall'occhio. L'ultimo passo consiste nell'applicare un algoritmo lossless ai dati rimanenti e il risultato finale è la codifica JPEG dell'immagine originale.
Nonostante i vantaggi dovuti alla semplicità e alla disponibilità di dispositivi hardware che realizzano la DCT, lo schema di compressione JPEG presenta delle limitazioni dovute al fatto che l'immagine viene elaborata a blocchetti, perciò la correlazione tra i bordi di blocchi adiacenti non viene eliminata. Questo provoca un fenomeno di *blocchettatura* dell'immagine.

In questa sezione si vuole introdurre una compressione di tipo lossy che è l'estensione dell'idea di Floater applicata alle immagini. Data un'immagine rappresentata da una matrice I, si prendono solo degli elementi appartenenti ad I e si applica uno schema di suddivisione bivariato approssimante visto nella sezione precedente. Lo schema si applica per diversi passi finchè si ha come risultato una matrice delle stesse dimensioni dell'immagine originale, si ricorda, infatti, che ad ogni passo una matrice NxM diventa di dimensioni $(2N-1)$x$(2M-1)$. Si ottiene, in questo modo, un'approssimazione dell'immagine originale. Si calcolano allora le differenze tra l'immagine esatta e la sua approssimazione e si considerano solo quelle maggiori di una tolleranza fissata. Si ha

quindi, invece che un'immagine, solo una piccola matrice di punti iniziali e una matrice rappresentante le differenze. Più in dettaglio, anche per il caso bivariato, si distinguono tre passaggi fondamentali, la decomposizione, la compressione e la decompressione.

Decomposizione
Sia I una matrice NxM che rappresenta un'immagine digitale. Gli elementi di I si indicano con $I(i,j)$. L'obiettivo è trovare una matrice $\hat{I}$ che approssimi I. Sia I_0 una matrice di dimensioni nxm che rappresenti una griglia della matrice iniziale I (la scelta di n e m e della matrice iniziale non è libera ma deve soddisfare alcuni requisiti che sono meglio spiegati nel capitolo sull'implementazione).
Sia R un procedimento di suddivisione bivariato che, data una matrice nxm, calcola il risultato del procedimento di suddivisione rappresentato da una matrice di dimensioni $(2n-1)$x$(2m-1)$.
Si applica a I_0 il procedimento di suddivisione bivariato R ottenendo I_1. Si itera questo procedimento fino ad ottenere, dopo k passi una matrice I_k di dimensioni NxM come la matrice originale.
Si definisce, ora, la matrice D delle differenze come

$$D := I - I_k$$

Si ottiene, in questo modo, la decomposizione I_0, D.

Compressione
Come per il caso di un segnale si fissa una tolleranza $\epsilon > 0$, si cerca l'approssimazione $\hat{D}$ che ha il numero minore possibile di elementi diversi da zero. Quindi si definisce:

$$\hat{D}(i,j) = \begin{cases} D(i,j), \text{ se } |D(i,j)| > \epsilon \\ 0, \text{ se } |D(i,j)| \leq \epsilon \end{cases} \qquad \text{per} i = 1, ...N \text{ e } j = 1, ..., M$$

La compressione dell'immagine, allora, consiste nel salvare non tutti gli elementi D ma solo gli elementi con errori significativi. La nuova matrice $\hat{D}$, sarà, infatti una matrice con un elevato numero di zeri e si può pensare di comprimerla con i metodi già noti in letteratura. Quindi, invece di salvare l'immagine I si salva solo I_0 e $\hat{D}$

Decompressione
Per decomprimere l'immagine si utilizza I_0 come matrice iniziale e si procede con il metodo di suddivisione bivariato R, ottenendo, dopo k passi la matrice I_k. A questo punto si aggiunge la matrice delle differenze $\hat{D}$ alla matrice appena calcolata ottenendo:

$$\hat{I} = I_k + \hat{D}$$

dove $\hat{I}$ rappresenta l'approssimazione dell'immagine originale cercata.

Questo nuovo metodo di compressione non risulta essere conveniente se non è supportato da un metodo che permetta di comprimere la matrice $\hat{D}$. Nei capitoli sull'implementazione e sulla sperimentazione sono riportati in dettaglio i problemi riscontrati.

Capitolo 3

Implementazione in Matlab

In questo capitolo si parlerà dell'implementazione Matlab dei vari schemi di suddivisione visti e delle loro applicazioni alle immagini. Si sono realizzate tre piccole interfacce grafiche che permettono all'utente di testare gli schemi di suddivisione, di provare ad ingrandire un'immagine utilizzando diverse tecniche e di comprimere delle immagini utilizzando algoritmi di suddivisione interpolanti. Inoltre è presente anche un codice che effettua la compressione di segnali. Di seguito sono spiegate tutte le problematiche relative all'implementazione degli schemi e sono presenti dei piccoli manuali che potrebbero servire ad utilizzare al meglio le interfacce.

3.1 Subdivision Curves

Per quanto riguarda le curve di suddivisione si sono realizzate diverse funzioni Matlab che implementano gli schemi di suddivisione per curve spiegati nel primo capiloto. Ogni funzione ha la stessa struttura: in input il vettore di lunghezza N dei punti su cui applicare lo schema e in output il nuovo vettore del risultato dello schema, di lunghezza $2N-1$. Si sono implementate diverse funzioni, sono divise tra funzioni che implementano gli schemi approssimanti e gli schemi interpolanti e di seguito sono dati i dettagli.

3.1. **Schemi di approssimazione**
Tra gli schemi di approssimazione implementati si hanno:

Linear Spline : schema B-spline lineare dell'esempio 1.1

Quadratic Spline : schema B-spline quadratico dell'esempio 1.2

Cubic Spline : schema B-spline cubico dell'esempio 1.3

Exponential : schema uniforme e non stazionario presentato nell'esempio 1.4. Questa funzione ha, in input anche il valore del parametro v^0 e si ricorda che:

- $v^0 \in (-1, 1) \Rightarrow$ lo schema riproduce le funzioni trigonometriche
- $v^0 = 1, \Rightarrow$ lo schema riproduce i polinomi in C^2
- $v^0 \in (1, \infty), \Rightarrow$ lo schema riproduce le funzioni iperboliche

Stat non Unif : schema stazionario non uniforme dell'esempio 1.5. Questa funzione ha, in input anche il valore del parametro ω relativo ad ogni lato.

Non Uniform 2 : schema non stazionario e non uniforme quadratico di Sabin esposto in [14], relativo all'esempio 1.6. Questa funzione ha in input i valori dei nodi relativi alla poligonale iniziale.

Non Uniform 3 : schema non stazionario e non uniforme cubico di Sabin dell'esempio 1.7. Come per lo schema quadratico, anche questa funzione ha bisogno della definizione dei nodi che serviranno per la definizione degli intervalli nodali.

3.2. **Schemi di interpolazione** Per quanto riguarda gli schemi di interpolazione si hanno:

N Point : schema a N punti dell'esempio 1.13, in input, questa funzione, richiede anche il numero di punti pari da utilizzare. Così se si fissa N=2 si ha il metodo dell'esempio 1.8, se N=4 si ha il classico schema a 4 punti dell'esempio 1.11, se N=6 si ha il metodo del'esempio 1.12. Non ci sono limitazioni alla scelta del numero N, in questo modo si può vedere il risultato degli schemi a 8,10 e 12 punti dell'esempio 1.13.

Four Point w : schema four point con parametro dell'esempio 1.14, la funzione in input ha anche il parametro ω che, si ricorda, deve essere $0 <= \omega <= 1/16$

Six Point w : schema six point con parametro ω dell'esempio 1.15. Anche in questa funzione si ha bisogno del valore del parametro ω che, per avere delle curve accettabili, deve essere tale che $0 <= \omega <= 1/16$

Circle 4pt : schema non stazionario uniforme del'esempio 1.17. Lo schema riproduce le circonferenze e funziona solo con curve chiuse che abbiano i punti periodici.

Non Stat 4pt : schema a 4 punti non stazionario uniforme dell'esempio 1.16. In input la funzione richiede il valore di v^0 ei ricorda che:

- $v^0 = 1 \Rightarrow$ lo schema riproduce i polinomi cubici
- $v^0 \in (0, +\infty)$, $s > 0 \Rightarrow$ lo schema riproduce le funzioni iperboliche
- $v^0 \in (-1, 1)$, $s > 0 \Rightarrow$ lo schema riproduce le funzioni trigonometriche

Variational : schema non stazionario uniforme che implementa il metodo variazionale di Kobbelt dell'esempio 1.18. Il metodo accetta solo poligonali chiuse.

4pt NULI : schema stazionario non uniforme che implementa il metodo nuli, spiegato nell'esempio 1.19. La funzione ha bisogno, in input, dei valori λ^0.

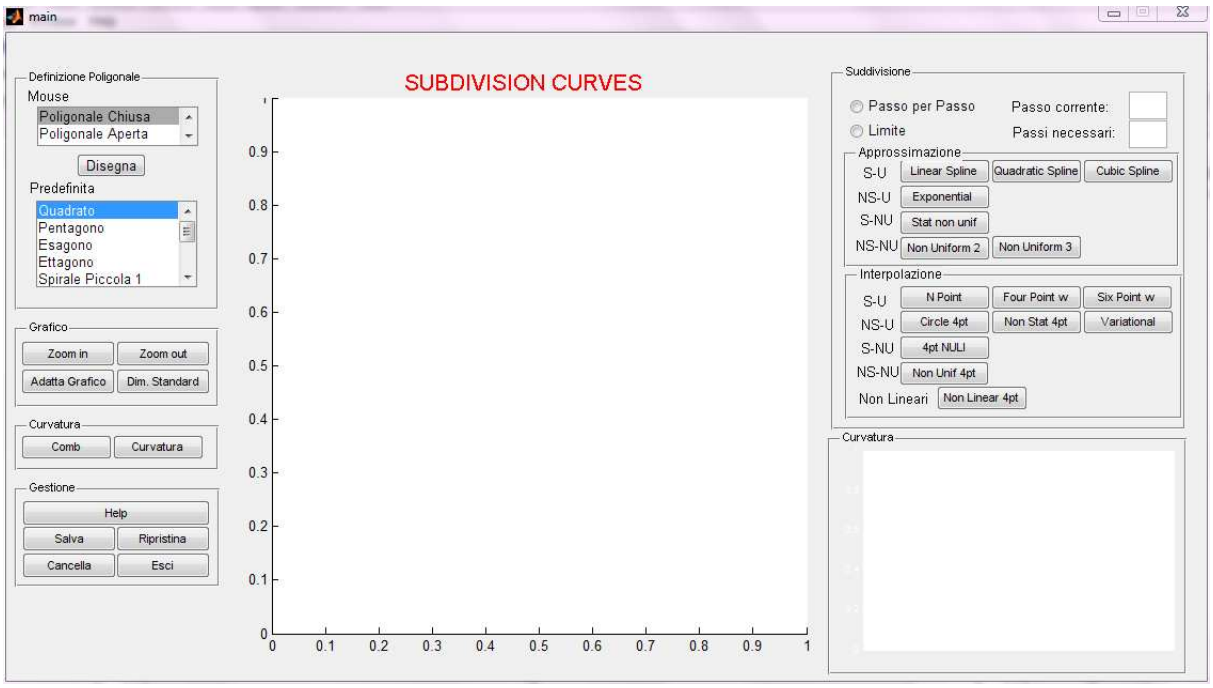
Non Unif 4pt : schema non stazionario e non uniforme relativo all'esempio 1.20. In input, bisogna immettere i valori dei pesi ω_i per ogni lato, si ricorda che, anche in questo caso, occorre che $0 <= \omega_i <= 1/16$.

Non Linear : schema non lineare dell'esempio 1.21. Anche in questo caso, la funzione richiede il valore del parametro di tensione ω.

Per utilizzare e confrontare tra loro tutte queste funzioni si è realizzata un'interfaccia grafica, spiegata meglio nel prossimo paragrafo.

Guida all'utilizzo dell'interfaccia

L'interfaccia grafica realizzata permette all'utente di osservare il comportamento dei diversi schemi di suddivisione realizzati. All'esecuzione del programma l'interfaccia si presenta come:



L'utente è chiamato a definire una poligonale iniziale nel riquadro in alto a sinistra. Si può disegnare direttamente nella finestra grafica una poligonale oppure si può scegliere una poligonale predefinita nel menù a tendina. Nel caso si scelga di disegnare una poligonale bisogna decidere se si vuole una poligonale aperta o chiusa, a quel punto cliccare

sul tasto `Disegna` e cliccare con il tasto sinistro del mouse nella finestra definendo le posizioni dei punti. Per terminare, in corrispondenza dell'ultimo punto, si clicca sul tasto destro del mouse. Nel caso di curve chiuse il primo e l'ultimo punto sono lo stesso e per terminare la poligonale iniziale bisogna cliccare con il destro in corrispondenza del penultimo punto. Degli esempi nella seguente figura:

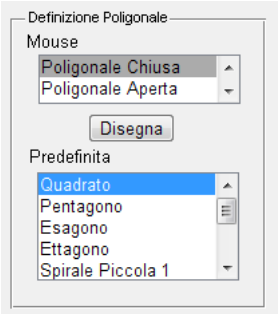


Figura 3.1: Definizione Poligonale

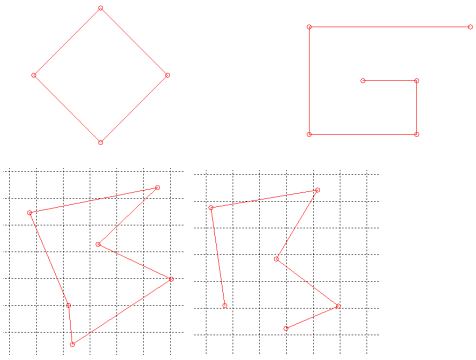
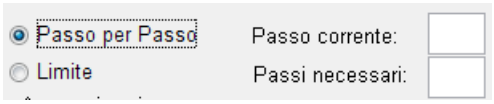


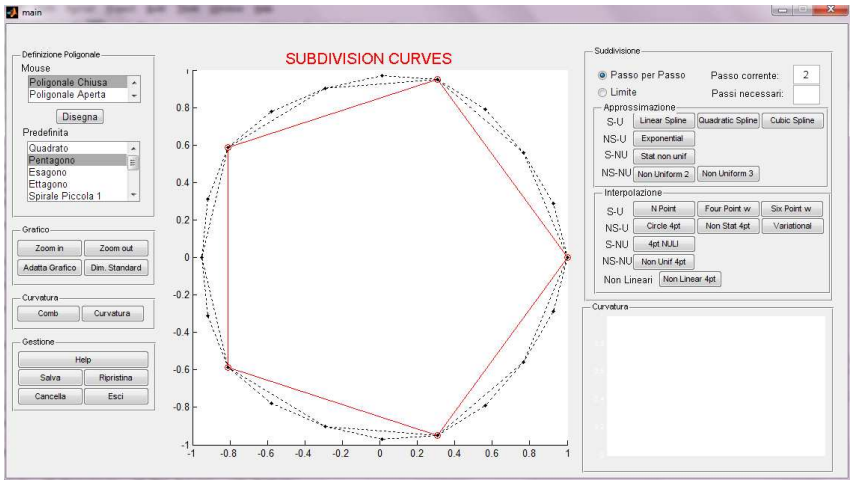
Figura 3.2: Sopra: curve predefinite chiuse e aperte. Sotto: esempio di definizione chiusa e aperta

Una volta definita la poligonale, per poter osservare il funzionamento dei metodi occorre scegliere se testare lo schema un passo alla volta, in questo caso si avrà solo un passo di suddivisione; oppure se vedere direttamente la curva limite del procedimento, scegliendo dai check box:

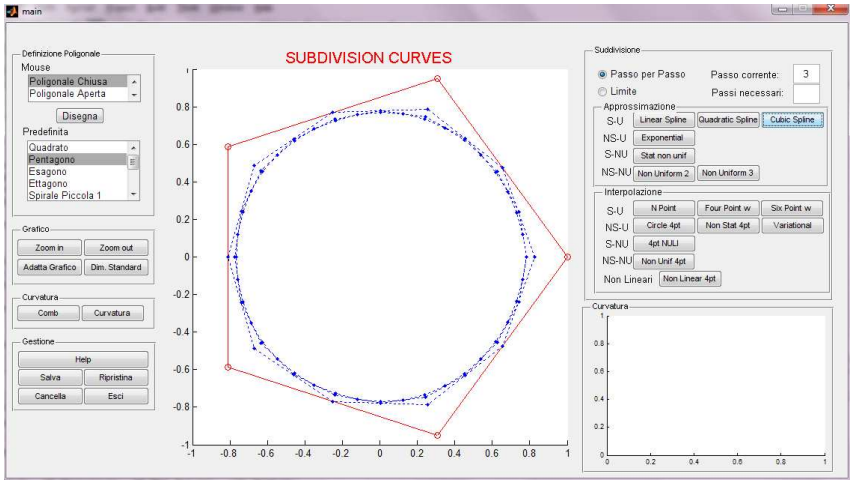


Nelle caselle di testo al lato si visualizza il passo corrente, se si è scelto il metodo passo per passo, oppure il numero dei passi necessari per arrivare alla curva limite, se, appunto si è scelto la curva limite.
Per curva limite non si intende la curva limite spiegata nel primo capitolo, in cui è rappresentata come la curva risultante quando il passo di suddivisione tende all'infinito, ma la curva ottenuta dopo diversi passi di suddivisione calcolati in modo tale che la distanza tra un valore e il successivo della curva non superi la grandezza di un pixel. Si può anche scegliere di osservare i primi passi di un procedimento di suddivisione e poi cambiare il check box e scegliere la curva limite. In questo caso i passi necessari per ottenere la curva limite visualizzati sono i passi necessari dal passo in cui si è effettuato il cambio in poi; quindi, se si vuole sapere il numero di passi necessari partendo dalla poligonale iniziale non bisogna far altro che sommare i passi delle due caselle si testo.

Comunque, una volta scelto il check box si può scegliere un metodo di suddivisione e si ottiene il seguente risultato:



Se si sceglie, come nella precedente figura, un metodo interpolante il risultato è visualizzato in nero, se, invece il metodo è approssimante, il risultato è in blu, come nella figura sotto.



Se si sceglie il metodo passo per passo sono evidenziati i nuovi punti e la nuova poligonale è tratteggiata, se, la scelta ricade sulla curva limite, la curva finale è rappresentata da una linea continua.
Si può scegliere tra i seguenti schemi, meglio spiegati nella prima parte di questa sezione.
Linear Spline: schema B-spline lineare dell'esempio 1.1
Quadratic Spline: schema B-spline quadratico dell'esempio 1.2
Cubic Spline: schema B-spline cubico dell'esempio 1.3
Exponential: schema presentato nell'esempio 1.4. Se si sceglie questo pulsante si aprirà una finestra di dialogo in cui verrà chiesto all'utente di scegliere il valore del parametro v^0.
Stat non Unif: schema dell'esempio 1.5. Anche in questo caso si aprirà una finestra di dialogo in cui l'utente dovrà immettere tali valori per ogni lato; il lato interessato è evidenziato.
Non Uniform 2: schema dell'esempio 1.6. Si chiedono all'utente i valori dei nodi.
Non Uniform 3: schema dell'esempio 1.7. Come per lo schema quadratico, si chiede all'utente di immettere il valore dei nodi.

N Point: schema a N punti dell'esempio 1.13. Si chiede all'utente il numero dei punti che definiranno le schema.
Four Point w: schema a 4 punti con parametro dell'esempio 1.14 in cui si chiede all'utente, con una finestra di dialogo, di inserire il parametro ω.
Six Point w: schema a 6 punti con parametro w dell'esempio 1.15. Anche in questo caso l'utente è chiamato a scegliere il parametro ω.
Circle 4pt: schema dell'esempio 1.17. Lo schema funziona solo con curve chiuse che abbiano i punti periodici. Se si vogliono vedere le funzionalità dello schema bisogna scegliere, dal menù a tendina, una tra le seguenti curve: `quadrato, pentagono, esagono o ettagono.`
Non Stat 4pt: schema dell'esempio **??**. Si permette all'utente, con una finestra di dialogo, di scegliere il parametro v^0.
Variational: schema che implementa il metodo variazionale di Kobbelt dell'esempio 1.18. Il metodo accetta solo poligonali chiuse; se si prova su una poligonale aperta una finestra avverte dell'errore.
4pt NULI: schema dell'esempio 1.19. Vengono chiesti all'utente con una finestra di dialogo, i valori del parametro λ. Lo schema necessita di tanti valori quanti sono i lati della poligonale iniziale; ogni volta che l'utente immette uno di questi valori il lato corrispondente è evidenziato.
Non Unif 4pt: schema dell'esempio 1.20. Si chiede all'utente di scegliere i valori dei pesi ω_i per ogni lato.
Non Linear: schema dell'esempio 1.21. Si chiede all'utente di scegliere il valore del parametro di tensione ω.
Si considerino, ora, i pulsanti apparteneti al gruppo *curvatura*. Si permette all'utente di

vedere la curvatura della curva presa in considerazione. Cliccando sul bottone `Curvatura` si ha che nell'area grafica riportato nella precedente figura si può visualizzare il grafico della curvatura. Se si clicca sul pulsante `Comb` si può vedere direttamente la curvatura sulla curva stessa, come si vede meglio dalle figure sottostanti.

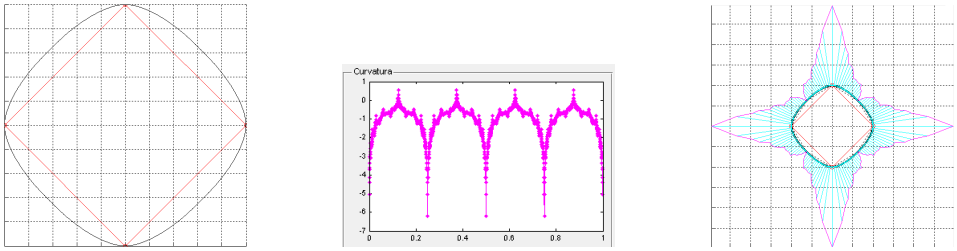
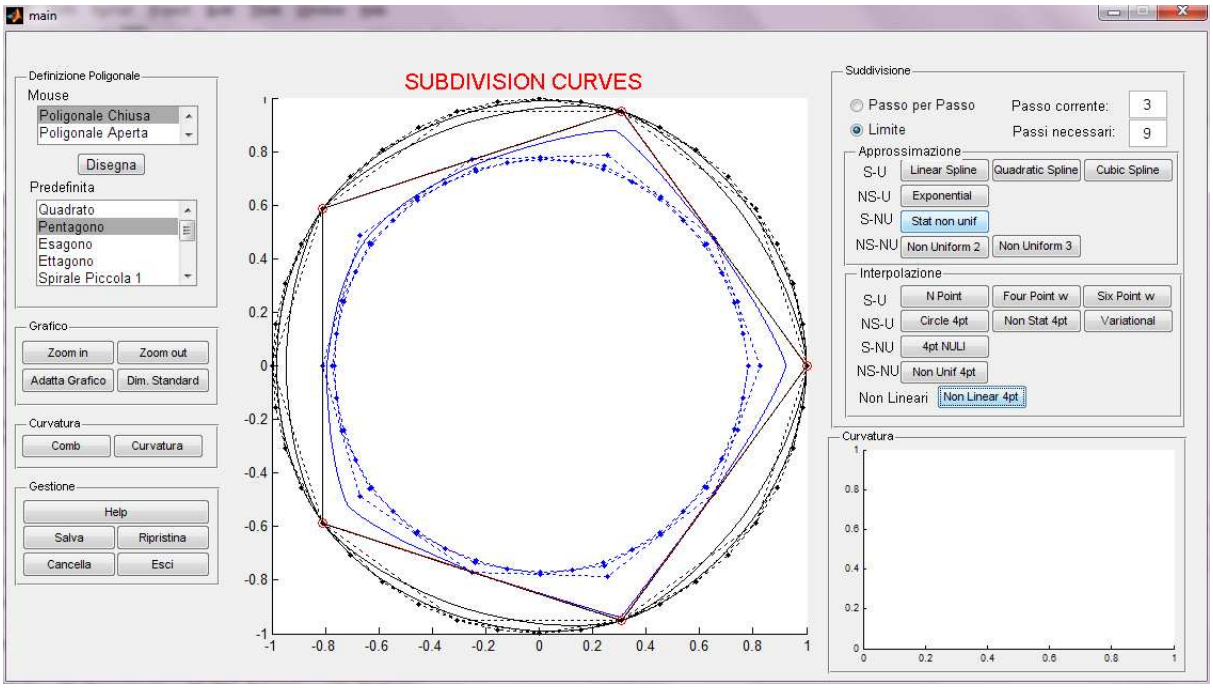


Figura 3.3: Esempio di schema applicato ad un quadrato, Curvatura e Curvatura sulla curva ottenuta con il tasto `Comb`

Invece, osservando il gruppo di bottoni relativi al grafico si ha che:
Zoom In: si permette all'utente di scegliere un punto nel grafico e la figura si ingrandisce considerando quel punto come il centro
Zoom Out: il grafico si rimpicciolisce di 1/2 ogni volta che si clicca
Adatta Grafico: si adattano le dimensioni del grafico alla figura
Dim. Standard: si torna alle dimensioni standard: [-1 1]x[-1 1]

Poi, osservando il gruppo relativo alla gestione si ha che:
Help: si ha una piccola guida all'uso
Salva: salva la poligonale iniziale in un file
Ripristina: ripristina ciò che era stato precedentemente salvato
Cancella: cancella i grafici
Esci: si esce dal programma
Infine, si vuole informare l'utente che si possono confrontare metodi diversi. Per fare ciò è necessario o utilizzare una curva predefinita oppure definire una nuova curva e salvarla. Si sceglie, allora, una curva e si applica il metodo scelto, si clicca di nuovo su quella curva predefinita oppure si ripristina la curva precedentemente salvata e si applica un altro metodo. Si può utilizzare questa tecnica tutte le volte che l'utente desidera. Nella seguente figura si mostra un esempio:



Un importante precisazione consiste nel fatto che se si vuole vedere la curvatura oppure il numero di passi o il risultato della funzione `Comb`, le applicazioni fanno riferimento all'ultima poligonale a cui è stato applicato qualche metodo.

3.2 Scaling

Si sono costruiti diversi codici Matlab che implementano i metodi di suddivisione bivariati e che li utilizzano per effettuare lo scaling di un'immagine digitale. Come spiegato nel precedente capitolo per effettuare l'ingrandimento di un'immagine digitale basta applicare un procedimento di suddivisione bivariato. Si sono implementate diverse funzioni Matlab che permettono di calcolare il risultato di uno schema di suddivisione bivariato. In particolare tutte le funzioni realizzate hanno in input la matrice da utilizzare per la suddivisione e, in output, la nuova matrice calcolata. Quindi se si vogliono più passi di suddivisione bisogna richiamare la stessa funzione più volte. Come spiegato nella Sezione 2.1 si hanno due definizioni differenti degli schemi di suddivione bivariati che dipendono dalla proprietà di approssimare o di interpolare che hanno gli schemi univariati da cui derivano. Si sono quindi implementati diversi esempi di schemi bivariati, alcuni approssimanti ed alcuni interpolanti.

3.3. **Schemi bivariati approssimanti**
Per questo categoria di schemi si è seguito sempre lo stesso procedimento. Si definisce la matrice maschera **M** e si calcola la nuova matrice con un procedimento di convoluzione. In particolare si sono implementati gli schemi:

B-spline lineare dell'esempio 2.1

B-spline quadratica dell'esempio 2.2

B-spline cubica dell'esempio 2.3

3.4. **Schemi bivariati interpolanti**
Per quanto riguarda gli schemi di interpolazione si sono implementati dei metodi che utilizzano nella definizione il prodotto tensoriale e un metodo che utilizza l'idea di Liang, Zou, Zhao e Qiu in [20]. In particolare si sono implementate le seguenti funzioni:

Four Point with w che rappresenta il schema bivariato dato dallo schema univariato del 4 punti con parametro dell'esempio 1.14. La definizione è analoga a quella dell'esempio 2.5 solo che ha il parametro ω nei coefficienti. Nella funzione realizzata, oltre alla matrice, si ha in input, anche il valore del parametro ω

N Point che rappresenta lo schema bivariato derivante da un qualsiasi schema univariato dell'esempio 1.13. L'implementazione di questo schema è come quella del metodo a 4 punti dell'esempio 2.4 o 2.5 ma, a seconda dello schema che si vuole ottenere, si ha una diversa definizione del prodotto tensoriale e degli elementi sulle righe e sulle colonne. Per essere più precisi si è implementata una funzione che, dato N un numero di punti pari, calcola la maschera dello schema univariato a N punti corrispondente. Questa funzione è richiamata quando si cercano i nuovi punti sulle riche e sulle colonne, in modo da avere N punti che definiscono il punto

cercato. É richiamata anche al momento di calcolare i nuovi punti con il prodotto tensoriale in modo da avere N^2 elementi della matrice originale che definiscono il nuovo punto. Con la funzione che calcola il risultato dello schema bivariato a N punti non si hanno limiti nella definizione di N; perciò si può vedere il risultato della suddivisione utilizzando degli N grandi.

Non Linear che rappresenta lo schema bivariato dell'esempio 2.6. Per questo schema si è utilizzato lo schema univariato dell'esempio 1.21. La funzione realizzata ha, in input, anche il valore del parametro di tensione ω

Nell'interfaccia realizzata si calcola lo zoom-in x2, x4 e x8 di un'immagine. Per calcolare lo zoom x2 basta richiamare una volta la funzione che rappresenta il metodo di suddivisione scelto. Per lo zoom x4 si applica il procedimento alla matrice risultante dal passo precedente, quindi la stessa funzione è chiamata due volte. Per lo zoom x8 si applica il applica il procedimento alla matrice risultante dallo zoom x4. In totale si calcolano tre passi di suddivisione.
Si procede quindi con la spiegazione dell'interfaccia.

Guida all'utilizzo dell'interfaccia

Si è costruito una semplice interfaccia che permette di utilizzare i codici spiegati in questa sezione. I programmi realizzati permettono di ingrandire di due, quattro e otto volte un'immagine digitale. Per ragioni di spazio si ingrandiscono immagini piccole, di dimensione 100 x 100 pixel. L'utente, avendo a disposizione diverse immagini grandi, è chiamato a scegliere una porzione di esse da ingrandire. L'utente può scegliere, inoltre, per effettuare lo scaling, i diversi metodi presentati in questa sezione. All'esecuzione del programma l'intefaccia si presenta come:

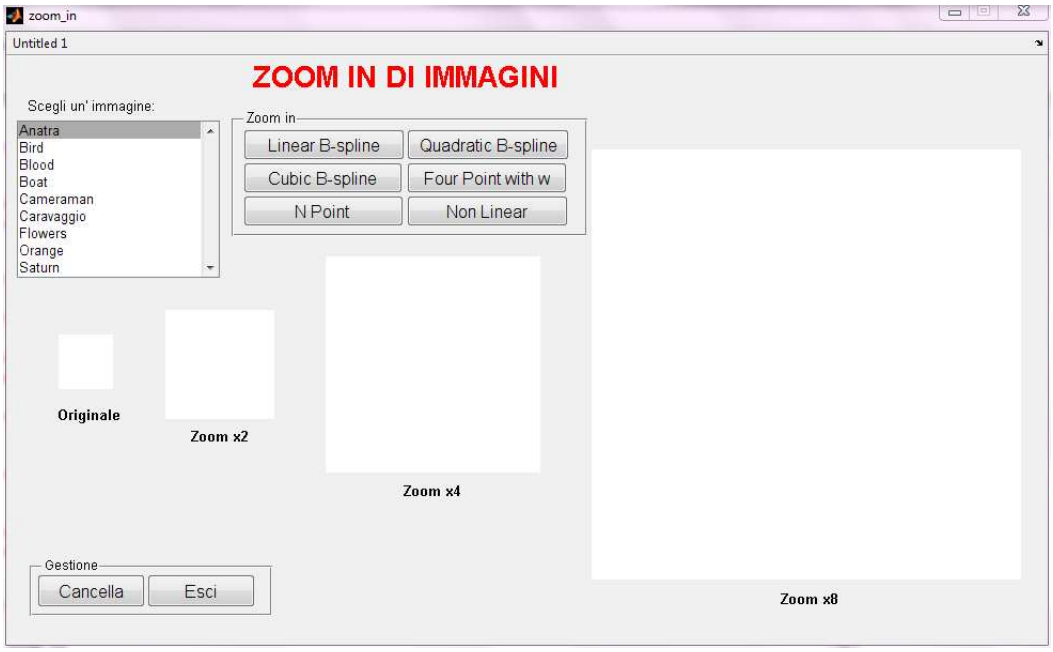
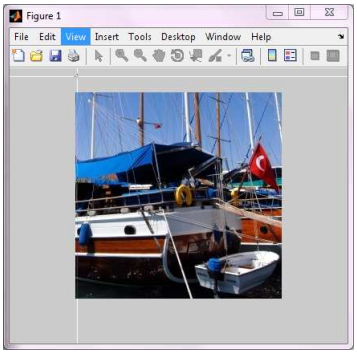
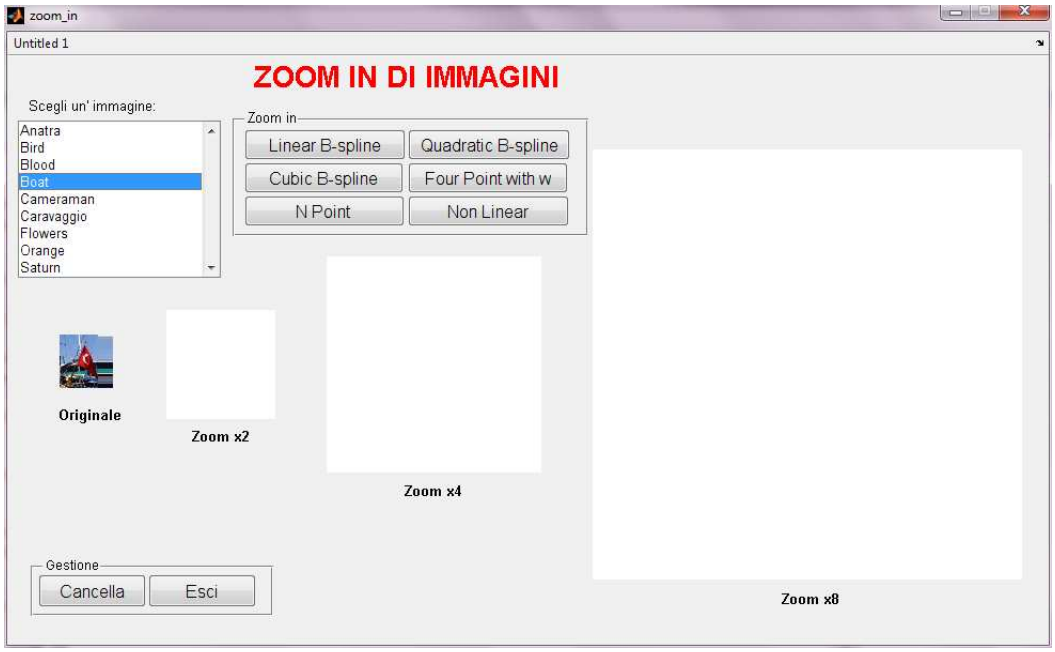


Figura 3.4:

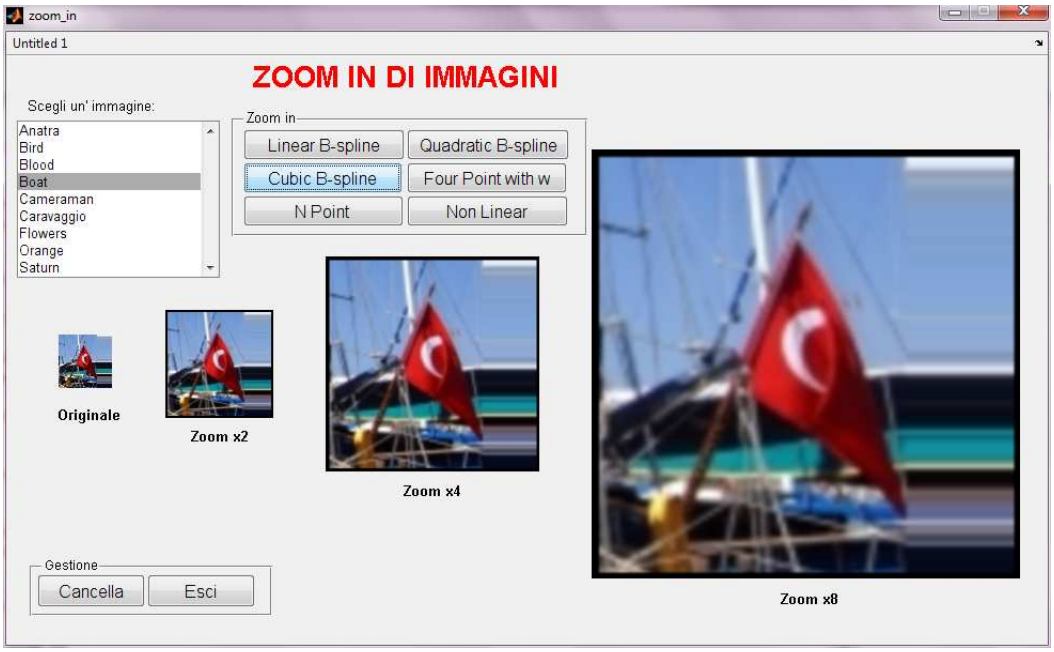
L'utente deve scegliere un'immagine dal menù a tendina sulla sinistra e, una volta effettuata la scelta, si visualizzerà una figura nel lato sinistro dello schermo con l'immagine originale, un esempio in figura 3.2.



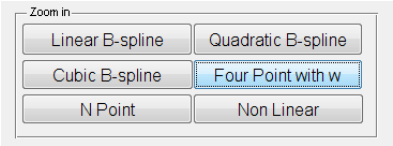
Si chiede, allora, di scegliere la porzione di immagine da ingrandire, l'utente deve cliccare su un punto dell'immagine visualizzata che sarà il centro della porzione di 100x100 pixel che il programma ingrandirà. Una volta scelto tale punto la finestra si chiude e, tale porzione appare nella finestra principale.



Ora l'utente può scegliere uno dei metodi presenti nel gruppo di bottoni e si ottiene il seguente risultato:



In questa schermata ci sono tre differenti nuove immagini, la prima rappresenta l'ingrandimento per due della porzione di immagine originale, la seconda l'ingrandimento per 4 e la terza l'ingrandimento per 8 della porzione di immagine scelta.
In particolare, osservando il gruppo di bottoni relativi ai metodi, si ha che:



Linear B-spline schema lineare dell'esempio 2.1
Quadratic B-spline schema quadratico dell'esempio 2.2
Cubic B-spline schema cubico di approssimazione dell'esempio 2.3
Four Point with w schema a quattro punti dell'esempio 2.5, si chiede all'utente, con una finestra di dialogo, di specificare il valore del parametro ω
N Point schema a N punti bivariato. Alla scelta di tale bottone, verrà chiesto, all'utente, di stabilire il numero di punti; che deve essere pari.
Non Linear schema non lineare dell'esempio 2.6 e, anche in questo caso, si chiede il valore del parametro ω.

Infine, i tasti `Cancella` ed `Esci` permettono di cancellare le immagini e di uscire dal programma. Se l'utente vuole vedere l'effetto di più schemi diversi sulla stessa porzione di immagine deve semplicemente cliccare su un'altro bottone del gruppo dei metodi.

3.3 Compressione

In questa sezione si parlerà dei programmi realizzati per la compressione. Nella prima parte si parlerà di compressione di segnali mentre nella seconda parte di compressione di immagini digitali per le quali è stata realizzata un'interfaccia grafica.

3.3.1 Compressione di Segnali

Si è realizzato un codice che permette di comprimere un segnale digitale. Si utilizza una funzione continua nota, come il sin o il cos, e si definisce un vettore I con una discretizzazione della stessa. Per far in modo che il risultato sia ottimale, bisogna prendere il vettore I di dimensione $2^k + 1$ per $k \geq 1$. L'obiettivo di questo codice è comprimere la discretizzazione del segnale, si definisce, allora il numero np dei punti che si vogliono memorizzare come punti iniziali. Di solito np è dispari e si definisce il nuovo vettore iniziale P^0 np-dimensionale con i punti equidistanti. Si fissa una tolleranza ϵ piccola a piacere e si definisce un vettore D di dimensione $2^k + 1$ che conterrà le differenze.
Si definisce anche lo schema di suddivisione interpolante utilizzare; si procede con l'applicazione dello stesso al vettore iniziale P^0 e si aggiorna il vettore delle differenze. Si itera questo procedimento (applicazione dello schema al vettore e aggiornamento del vettore delle differenze) finchè il vettore P^0 non diventa di dimensione $2^k + 1$.
A questo punto si finisce il primo passaggio del metodo di compressione, cioè si è appena effettuata la decomposizione. Ora bisogna comprimere il vettore delle differenze; si osserva semplicemente un valore per volta, se tale valore è maggiore di ϵ si lascia invariato, invece, se il valore è minore o uguale si fissa a zero. Il codice, a questo punto, fornisce all'utente il numero e la percentuale degli zeri contenuti nel vettore D.
Infine si decomprime il segnale applicando il metodo di suddivisione al vettore P^0 iniziale e si aggiunge il vettore delle differenze D. Per una spiegazione grafica si veda il seguente esempio.

Esempio 3.1. In questo esempio si vuole far vedere come funziona il codice sopra descritto.
Decomposizione Sia k=3, si definisce, quindi il vettore della discretizzazione I di lunghezza 9.

$$\text{I}= \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid P_5 \mid P_6 \mid P_7 \mid P_8 \mid P_9 \mid$$

sia $np = 3$ allora $P^0 = \mid P_1 \mid P_5 \mid P_9 \mid$Si fissa una tolleranza ϵ e definisce anche il vettore delle differenze D di lunghezza 9

$$\text{D}= \mid 0 \mid \cdot \mid \cdot \mid \cdot \mid 0 \mid \cdot \mid \cdot \mid \cdot \mid 0 \mid$$

alle posizioni 1,5 e 9 è nullo perché rappresenta la differenza tra i valori di I esatti e i valori salvati in P^0 che, quindi, è nulla. Poi si sceglie lo schema di suddivisione interpolante

con cui voler provare la compressione e si inizia ad applicare tale schema al vettore P^0.
Primo passo di suddivisione
Applicando lo schema al vettore P^0 si ottiene:

$$P^0 = \mid P_1 \mid \hat{P}_3 \mid P_5 \mid \hat{P}_7 \mid P_9 \mid$$

dove $\hat{P}_i$ sono i nuovi valori calcolati e rappresentano l'approssimazione dei valori con lo stesso indice nel vettore esatto I. Infatti si aggiorna il vettore delle differenze come:

$$\mathrm{D} = \mid 0 \mid \cdot \mid e_3 \mid \cdot \mid 0 \mid \cdot \mid e_7 \mid \cdot \mid 0 \mid$$

con $e_i = \left\| P_i - \hat{P}_i \right\|$
Secondo passo di suddivisione
Applicando di nuovo lo schema al vettore P^0 si ha:

$$P^0 = \mid P_1 \mid \hat{P}_2^1 \mid \hat{P}_3 \mid \hat{P}_4^1 \mid P_5 \mid \hat{P}_6^1 \mid \hat{P}_7 \mid \hat{P}_8^1 \mid P_9 \mid$$

dove $\hat{P}_i^1$ sono i nuovi valori calcolati, si aggiorna, quindi, il vettore delle differenze:

$$\mathrm{D} = \mid 0 \mid e_2 \mid e_3 \mid e_4 \mid 0 \mid e_6 \mid e_7 \mid e_8 \mid 0 \mid$$

con $e_i = \left\| P_i - \hat{P}_i^1 \right\|$
Compressione Si aggiorna il vettore D come:

$$\mathrm{D} = \mid 0 \mid \hat{e}_2 \mid \hat{e}_3 \mid \hat{e}_4 \mid 0 \mid \hat{e}_6 \mid \hat{e}_7 \mid \hat{e}_8 \mid 0 \mid$$

con

$$\hat{e}_i = \begin{cases} e_i \text{ se } e_i > \epsilon \\ 0 \text{ se } e_i \leq \epsilon \end{cases}$$

Decompressione Partendo dal vettore P^0 si applica due volte lo schema di suddivisione scelto. Passo iniziale: $P^0 = \mid P_1 \mid P_5 \mid P_9 \mid$
Passo 1: $P^0 = \mid P_1 \mid P_3 \mid P_5 \mid P_7 \mid P_9 \mid$
Passo 2: $P^0 = \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid P_5 \mid P_6 \mid P_7 \mid P_8^1 \mid P_9 \mid$ a cui si aggiunge il vettore D
$\mathrm{D} = \mid 0 \mid \hat{e}_2 \mid \hat{e}_3 \mid \hat{e}_4 \mid 0 \mid \hat{e}_6 \mid \hat{e}_7 \mid \hat{e}_8 \mid 0 \mid$
Si ottiene, allora, un'approssimazione del vettore iniziale I data da
$P^0 = \mid P_1 \mid P_2 + \hat{e}_2 \mid P_3 + \hat{e}_3 \mid P_4 + \hat{e}_4 \mid P_5 \mid P_6 + \hat{e}_6 \mid P_7 + \hat{e}_7 \mid P_8 + \hat{e}_8 \mid P_9 \mid$

3.3.2 Compressione di Immagini

In questa sezione si vuole spostare l'attenzione sulla compressione di immagini digitali. Per comprire le immagini si utilizzano i metodi di suddivisione in due dimensioni, utilizzati anche per lo scaling. In questa particolare applicazione sono utilizzati solo i metodi di interpolazione per come è strutturato il problema. Quindi si utilizzano le funzioni implementate per il metodo a 4 punti con parametro, il metodo a N punti e il metodo non lineare tangent-driven. Sia I l'immagine originale, e, come per i segnali, sia di dimensione $(2^k + 1)$x$(2^h + 1)$. Per semplicità si sono implementati codici solo per immagini quadrate.
Il procedimento è lo stesso utilizzato per i segnali ma, ora, si lavora, invece che con vettori, con matrici. Sia np il numero di elementi che si vogliono utilizzare per la matrice iniziale allora si definisce una nuova matrice P0 di dimensione npxnp con gli elementi di I equispaziati. Simbolicamente:

$$I = \begin{bmatrix} \bullet & \cdot & \bullet & \cdot & \bullet \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \bullet & \cdot & \bullet & \cdot & \bullet \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \bullet & \cdot & \bullet & \cdot & \bullet \end{bmatrix} \Rightarrow P0 = \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix}$$

Come per i segnali, si definisce la matrice delle differenze D e si inizia ad applicare uno schema di suddivisione bivariato. Ad ogni passo si aggiorna anche la matrice delle differenze e si continua ad applicare il metodo di suddivione finchè la matrice P0 non ha dimensione uguale alla dimensione di I. A questo punto si aggiorna la matrice delle differenze assegnando il valore nullo a quegli elementi che sono minori di una tolleranza fissata. Il codice allora fornisce il numero e la percentuale degli zeri.
La seconda parte del codice riguarda la ricostruzione dell'immagine. Si parte dalla matrice iniziale e si applica lo schema di suddivisione finchè P0 non diventa di dimensioni $(2^k + 1)$x$(2^k + 1)$. Si aggiunge poi la matrice delle differenze e si ha l'immagine ricostruita.
Si nota che per le immagini nella scala di grigio si lavora con una sola matrice delle differenze mentre per le immagini a colori si hanno tre matrici delle differenze, una per ogni colore.
Inoltre, per stabilire se l'immagine ricostruita è una buona riproduzione dell'immagine originale si introducono degli indicatori di qualità, quali: PSNR, Q e PEE.

3.5. **PSNR**
L'indicatore PSNR, peak signal-to-noise ratio, è definito come il rapporto tra la massima potenza di un'immagine e la potenza di rumore che può invalidare la fedeltà della sua rappresentanzione ricostruita. Prima di esaminare il PSNR è bene definire l'MSE, Mean

Square Error, come

$$MSE = \frac{1}{(nm)} \sum_{i=1}^{n} \sum_{j=1}^{m} \|I_0(i,j) - I_1(i,j)\|^2 \tag{3.1}$$

Il MSE fornisce un grado di similarità/fedeltà o un livello di errore/distorsione tra le immagini I_0 e I_1; se MSE=0 allora le immagini sono identiche. Si può ora definire il PSNR come:

$$PSNR = 20 \log_{10} \left(\frac{\max\{I\}}{\sqrt{MSE}} \right) \tag{3.2}$$

dove:

$$\max\{I\} = \begin{cases} 1, \text{ per immagini binary} \\ 255, \text{ per immagini a scala di grigio} \\ 255, \text{ per ogni matrice delle immagini a colori} \end{cases}$$

Il PSNR è solitamente espresso in termini di scala logaritmica di decibel e un incremento significativo per la percezione umana è di 0.25 dB. Il valore del PSNR tende all'infinito per il valore del MSE che tende a zero; questo significa che un più alto valore del PSNR corrisponde ad un'immagine di maggior qualità. Se, invece, il valore del PSNR è piccolo significa che le immagini confrontate sono molto diverse numericamente [27].
Se si considera un'immagine nella scala di grigio si avrà un unico valore del PSNR; se, invece, si considera un'immagine a colori si avranno tre diversi valori del PSNR, uno per ogni matrice, e il PSNR totale sarà dato da:

$$PSNR_{totale} = 20 \log_{10} \left(\frac{\max\{I\}}{\sqrt{\frac{MSE_R + MSE_G + MSE_B}{3nm}}} \right) \tag{3.3}$$

dove MSE_R, MSE_G e MSE_B sono i valori del MSE riferiti alla matrice dei rossi, dei verdi e dei blu.

3.6. Q

Il metodo della qualità dell'immagine Q, quality index, che misura la distorsione di un'immagine ricostruita rispetto all'immagine originale, è definito come:

$$Q = \frac{4\sigma_{I_0 I_1} \bar{I}_0 \bar{I}_1}{\left(\sigma_{I_0}^2 + \sigma_{I_1}^2\right) \left[\bar{I}_0{}^2 + \bar{I}_1{}^2\right]} \tag{3.4}$$

dove

$$\bar{I} = \sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j) \tag{3.5}$$

rappresenta la media dell'immagine I;

$$\sigma_I^2 = \frac{1}{(n-1)(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(I(i,j) - \bar{I}\right)^2 \tag{3.6}$$

rappresenta la varianza dell'immagine I e

$$\sigma_{I_0,I_1}^2 = \frac{1}{(n-1)(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(I_0(i,j) - \bar{I_0}\right) \left(I_1(i,j) - \bar{I_1}\right) \tag{3.7}$$

rappresenta la covarianza tra le immagini I_0 e I_1.
I valori del parametro Q variano nell'intervallo $[-1, 1]$. Il miglior valore, Q=1, si raggiunge solo se le due immagini sono uguali, cioè se $\forall i, j$ si ha che $I_0(i,j) = I_1(i,j)$; invece, il valore più basso, Q=-1, si ha quando $I_1(i,j) = 2\bar{I_0} - I_0(i,j)\ \forall i, j$.
Q è dato da tre diversi fattori Q_1, Q_2 e Q_3; che rappresentano rispettivamente la perdita di correlazione, la distorsione della luminosità e la distorsione del contrasto e sono definiti dalla seguente equazione:

$$Q = Q_1 Q_2 Q_3 = \frac{\sigma_{I_0 I_1}}{\sigma_{I_0}\sigma_{I_1}} \cdot \frac{2\bar{I_0}\bar{I_1}}{\bar{I_0}^2 + \bar{I_1}^2} \cdot \frac{2\sigma_{I_0}\sigma_{I_1}}{\sigma_{I_0}^2 + \sigma_{I_1}^2} \tag{3.8}$$

Q_1 è il coefficiente di correlazione tra I_0 e I_1 e misura il grado di correlazione lineare tra le due immagini. Il suo valore varia nell'intervallo $[-1, 1]$. Il miglior risultato si raggiunge se $Q_1 = 1$ e si ottiene quando $I_1(i,j) = aI_0(i,j) + b\ \forall i, j$ con $a > 0$ e b costanti. Però, anche se le due immagini risultano correlate si possono ancora trovare delle distorsioni tra loro, tali incongruenze sono valutate dal secondo e terzo fattore.
Q_2, infatti, confronta la luminosità delle due immagini ed ha come range di valori $[0, 1]$. Se $Q_2 = 0$ allora le immagini non hanno la stessa luminosità; infatti il miglior valore, $Q_2 = 1$, si raggiunge solo quando le due immagini sono uguali, quindi se hanno anche la stessa luminosità.
Infine, il terzo fattore Q_3 ha, anch'esso, range $[0, 1]$ e misura il contrasto tra I_0 e I_1. Il miglior risultato si ottiene quando $Q_3 = 1$ e ciò è vero solo se $\sigma_{I_0} = \sigma_{I_1}$; infatti i valori σ_{I_0} e σ_{I_1} possono essere visti come la misura del contrasto tra le immgini [26].
Anche per questo metodo se si considera un'immagine nella scala di grigio si avrà un unico valore; se, invece, si considera un'immagine a colori si avranno tre diversi valori del Q, uno per ogni matrice, e il Q totale, in questo caso sarà dato da:

$$Q_{totale} = \frac{Q_R + Q_G + Q_B}{3}$$

dove Q_R, Q_G e Q_B rappresentano i valori del Q per le matrici dei rossi, dei verdi e dei blu.

3.7. **PEE**

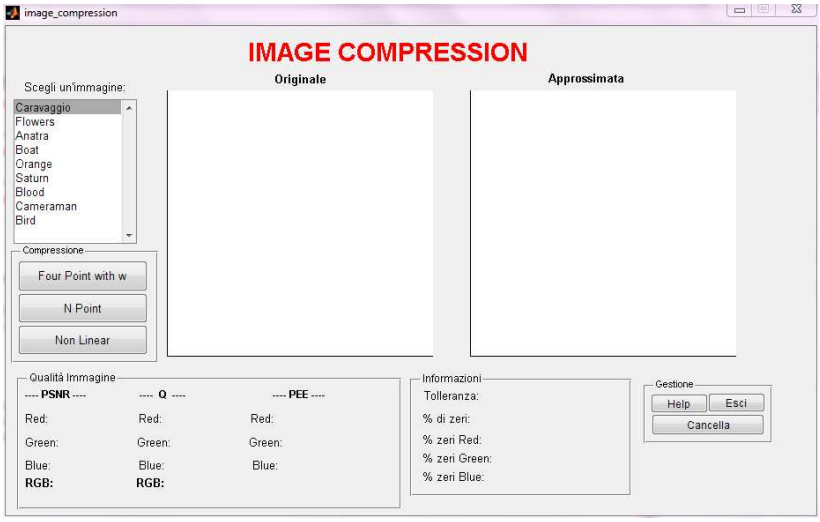
Il metodo PEE, Percentage Edge Error, introdotto da Al-Fahoum e Reza in [24] è definito come:

$$PEE = \frac{\bar{I}_0 - \bar{I}_1}{\bar{I}_0} \text{ x } 100\% \tag{3.9}$$

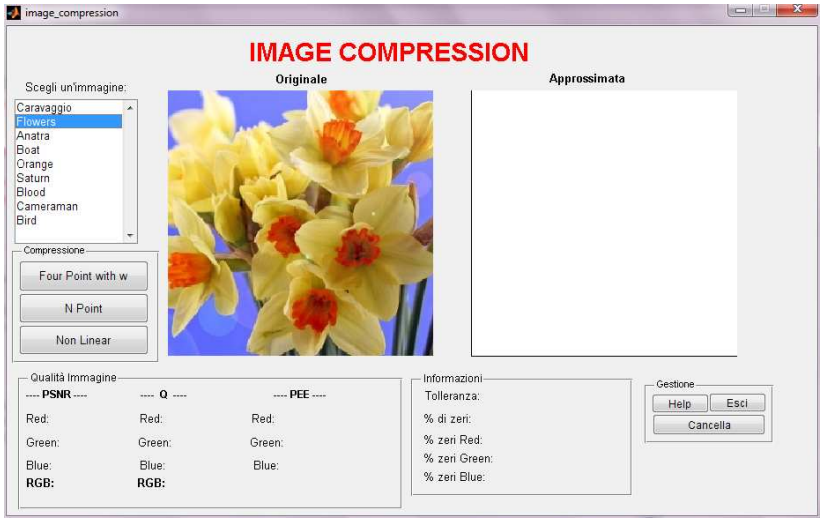
dove $\bar{I}$, è la media relativa all'immagine I definita nell'equazione 3.5 come $\bar{I} = \sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j)$. Questo indicatore di qualità misura la verosomiglianza dei dettagli delle immagini, cioè misura quanto i dettagli della nuova immagine sono vicini ai dettagli dell'immagine originale. Il PEE è una misura dell'insoddisfazione, dovuta agli artefatti, percepita da chi guarda l'immagine.
Se il PEE risulta essere negativo si ha che i dettagli dell'immagine ricostruita verosomiglianti con i dettagli dell'immagine originale ; al contrario, se il PEE è positivo l'immagine ricostruita risulta corrotta, ossia presenta diversi artefatti e i dettagli non sono verosimili.
Quindi la diminuzione del valore del PEE implica un miglioramento nella qualità dell'immagine ricostruita [24] [25].
Infine, se si considera un'immagine nella scala di grigio si avrà un unico valore del PEE; se, invece, si considera un'immagine a colori si avranno tre diversi valori del PEE, uno per ogni matrice. Al contrario degli altri metodi, non si ha un'espressione del PEE totale.

Guida all'utilizzo dell'interfaccia

È stata implementata un'interfaccia grafica che permette di comprimere le immagini che permette all'utente di osservare il risultato del processo di compressione spiegato nella prima parte di questa sezione. Si è pensato a quasto tipo di implementazione in modo che l'utente possa variare la tolleranza e utilizzare diversi metodi. Inoltre può visualizzare, ad ogni prova, i valori dei parametri di qualità relativi all'immagine ricostruita. Con questa interfaccia può rendersi conto dell'efficienza dei codici e degli algoritmi sopra descritti e valutare, di prova in prova, quale tipo di compressione risulta essere la più conveniente. Mandando in esecuzione il programma l'utente è chiamato ad interagire con la seguente interfaccia.

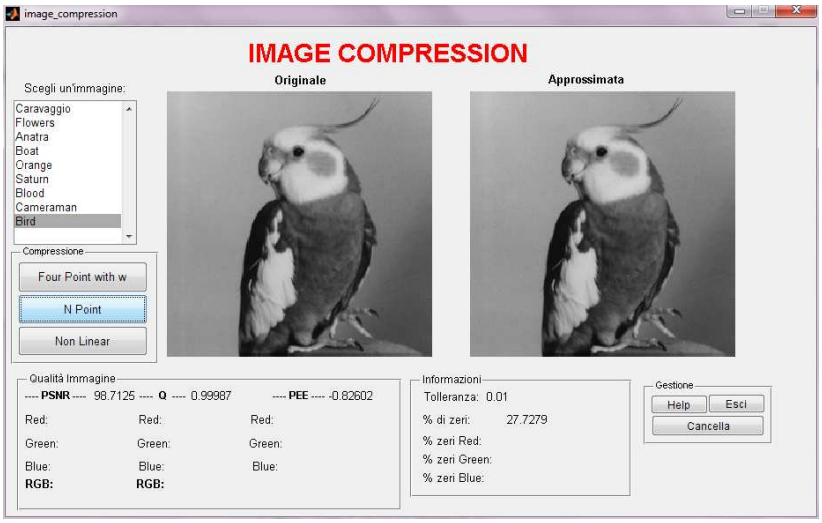


L'utente deve scegliere un'immagine dal menù a tendina presente sulla sinistra dell'interfaccia; tale immagine sarà visualizzata come segue:



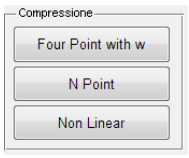
Ora l'utente può scegliere un metodo di compressione nel relativo gruppo di bottoni, verrà chiesto il valore della tolleranza a cui fare riferimento nella compressione attraverso una finestra di dialogo; ed il risultato finale sarà:

La nuova immagine visualizzata rappresenta la ricostruzione dell'immagine dopo la compressione realizzata con il metodo scelto; in basso sono presenti tutti i valori relativi alla qualità della nouva immagine messa in relazione con l'immagine originale. Sono quindi presenti, per quanto riguarda le immagini a colori, i valori del PSNR, Q, PEE per componente di colore e, sia del PSNR che del Q anche totale. Nel riquadro in basso a destra sono presenti sia il valore della tolleranza fissata al momento della scelta che la precentuale di zeri nelle matrici delle componenti di colore.
Le immagini in bianco e nero sono rappresentate da una sola matrice percui i valori relativi alla qualità e la percentuale degli zeri è riferita solo a quella matrice, come si nota nella seguente figura:



Infine si hanno i bottoni `Cancella` ed `Esci` che servono a pulire le finestre e ad uscire

dal programma.
Osservando meglio il gruppo di bottoni relativi ai metodi si ha che:



Four Point with w schema a quattro punti dell'esempio 2.5, si chiede all'utente, con una finestra di dialogo, di specificare il valore del parametro ω
N Point schema a N punti bivariato. Alla scelta di tale bottone, verrà chiesto, all'utente, di stabilire il numero di punti; che deve essere pari.
Non Linear schema non lineare dell'esempio 2.6, e, anche in questo caso, si chiede il valore del parametro ω.

Capitolo 4

Sperimentazione

In quest'ultimo capitolo sono riportati i risultati ottenuti utilizzando i programmi spiegati nel capitolo precedente e sono spiegate le conclusioni relative.

4.1 Subdivision Curves

In questa sezione sono spiegati degli esempi di schemi di suddivisione, i risultati sono ottenuti dai codici implementati. Gli schemi e gli esempi sono divisi in categorie come nel primo capitolo.

4.1. **Approssimazione Uniforme Stazionario**
In questa prima classificazione ci sono gli schemi basati sulle funzioni B-spline, in particolare il metodo lineare, dell'esempio 1.1; il metodo quadratico, dell'esempio 1.2 e il metodo cubico, dell'esempio 1.3. Come già spiegato lo schema lineare riproduce esattamente la poligonale iniziale. Il metodo quadratico anche detto *Corner-Cutting* ha la particolarità di *smussare gli angoli* e la curva limite calcolata rimane più vicino alla poligonale iniziale rispetto al metodo cubico. Se si prende la stessa poligonale iniziale, si ha che i metodi forniscono riportati nelle figure 4.1 e 4.1.

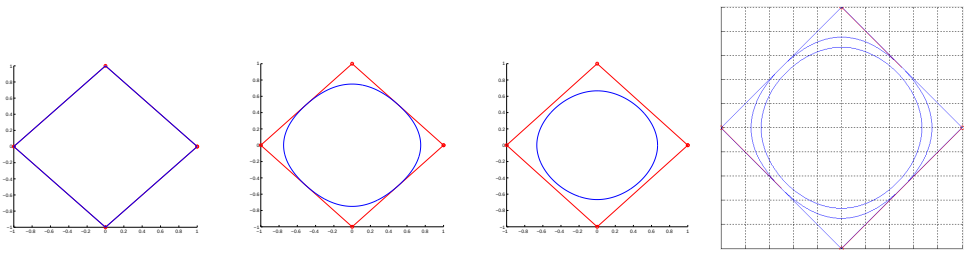


Figura 4.1: B-spline lineare; quadratica; cubica e sovrapposizione di tutte

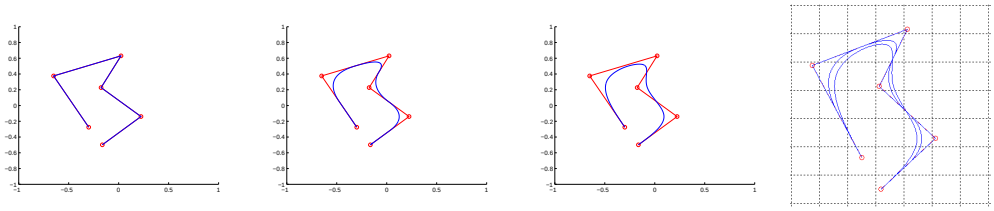


Figura 4.2: B-spline lineare; quadratica; cubica e sovrapposizione di tutte

4.2. **Approssimazione Uniforme Non Stazionario**

Lo schema rappresentante questa classe è quello che riproduce le B-spline esponenziali dell'esempio 1.4. Si ricorda che si possono fissare diversi valori del parametro v^0, ottenendo diverse curve limite. Alcuni esempi possono essere osservati in figura 4.3.

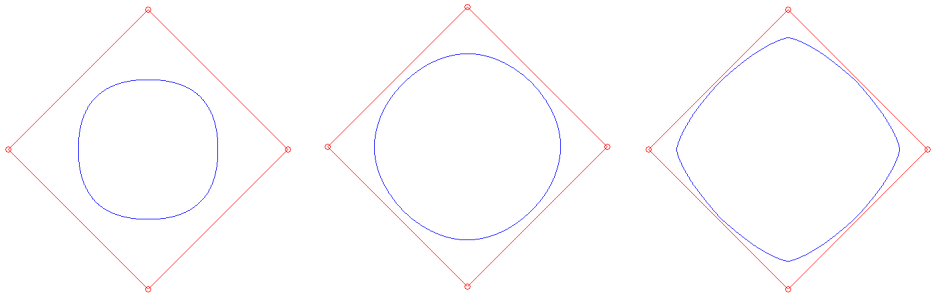


Figura 4.3: Exponential B-spline per $v^0 = 0$; $v^0 = 1$ e $v^0 = 3$

Nelle immagini in figura 4.4 si sovrappongono i risultati dello schema per diversi valori di v^0. Osservando le prime due figure si ha che $v^0 = 0$, lo schema riproduce le funzioni trigonometriche e si ha la curva più lontana dalla poligonale iniziale, per $v^0 = 1$ il metodo riproduce i polinomi in C^2 e si ha la curva intermedia e per $v^0 = 5$ lo schema riproduce le funzioni iperboliche e si ha la curva più vicina alla poligonale iniziale. Si può, inoltre notare lo stesso comportamento del limite osservando la terza figura in cui $v^0 = 2, 5, 10$. Si vede, allora, che aumentando il valore del parametro v^0 si hanno delle curve limite molto vicine alla poligonale iniziale

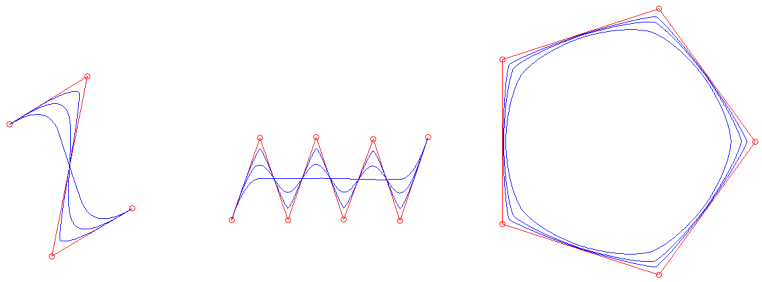


Figura 4.4: Exponential B-spline per $v^0 = 0, 1, 5$; $v^0 = 0, 1, 5$ e $v^0 = 2, 5, 10$

4.3. **Approssimazione Non Uniforme Stazionario**
Lo schema presentato nell'esempio 1.5 è non uniforme e stazionario. La maschera relativa è data da: $a := \left\{\frac{\omega_i}{2}, \frac{1}{2}, (1-\omega_i), \frac{1}{2}, \frac{\omega_i}{2}\right\}$; quindi si possono fissare diversi valori del parametro ω per ogni lato.

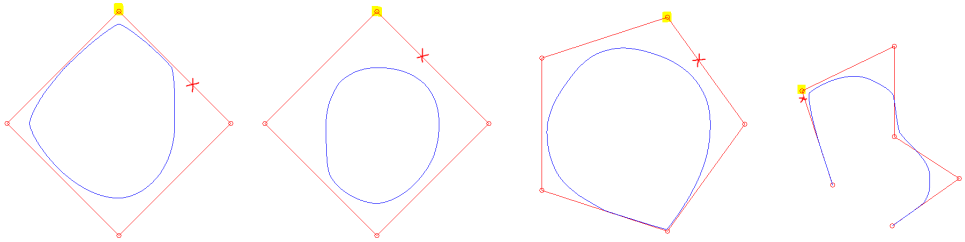


Figura 4.5: Approssimazione non uniforme stazionaria: $\omega_i = (0.0625, 0.125, 0.25, 0.5)$; $\omega_i = (0.5, 0.6, 0.2, 0.4)$; $\omega_i = (0.5, 0.4, 0.2, 0.01, 0.4)$ e $\omega_i = (0.0625, 0.5, 0.1, 0.4, 0.0625)$

In figura 4.5 i valori dei parametri si riferiscono ai lati, partendo dal lato evidenziato in rosso e proseguendo in senso antiorario per le curve chiuse. Si nota che se ω_i risulta essere grande la curva limite tende ad allontanarsi dai vertici relativi ai lati i-esimi partendo dal vertice evidenziato in giallo); se invece ω_i è piccolo, la curva finale è molto vicina al vertice di riferimento.
Si nota, inoltre, che se si fissano tutti gli $\omega_i = 0$ o $\omega_i = \frac{1}{4}$ lo schema è esattamente il B-spline lineare e B-spline cubico, come mostrato in figura 4.6.

4.4. **Approssimazione Non Uniforme Stazionario**
In questa classe rientrano gli schemi presentati negli esempi 1.6 e 1.7. Si ricorda che, per questi metodi, c'è bisogno di fissare il valore dell'intervallo nodale per ogni vertice, se si tratta dello schema quadratico, o per ogni lato, se si tratta delllo schema cubico. A seconda di come si definiscono gli intervalli, si hanno diversi risultati della curva limite. Se i valori degli intervalli nodali sono tutti uguali ad 1 si ottengono esattamente i metodi B-spline quadratico e cubico.
Per quanto riguarda il metodo quadratico si ha che la curva limite è più o meno vicino

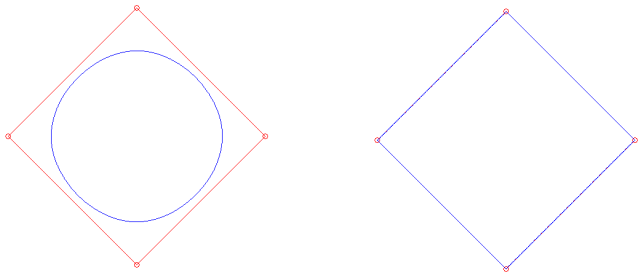


Figura 4.6: Approssimazione non uniforme stazionaria: $\omega_i \equiv 0.25$ B-spline cubiche; $\omega_i \equiv 0$ B-spline lineari

ai vertici della poligonale iniziale a seconda del valore dell'intervallo nodale definito. In particolare più l'intevallo è alto più la curva limite di allontanerà dalla poligonale; se, invece, il valore risulta piccolo, il limite sarà molto vicino ai vertici fino ad interpolarli se il valore d_i risulta essere nullo. Nelle figure 4.7 ci sono alcuni esempi e nella didascalia sono riportati i valori degli intervalli nodali riferiti ai vertici, partendo dal vertice evidenziato in giallo e proseguendo in modo antiorario.

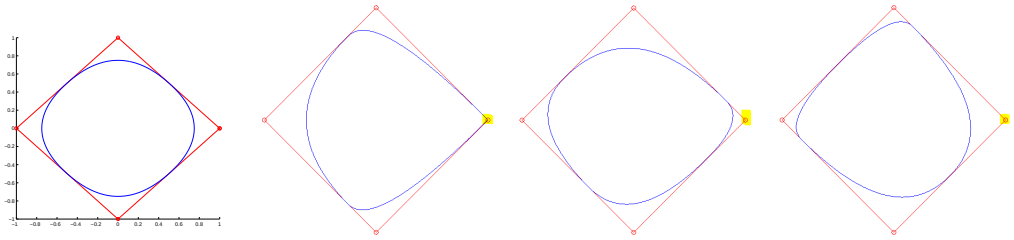


Figura 4.7: Approssimazione non uniforme non stazionaria quadratica: B-spline quadratica ($d_i \equiv 1$); $d_i = (0, 1, 3, 1)$; $d_i = (1, 5, 3, 2)$ e $d_i = (5, 1, 1, 5)$

Lo schema cubico ha gli intervalli nodali riferiti ai lati. Osservando anche gli esempi proposti nelle figure 4.8, si nota che la curva limite si avvicina ai lati a seconda del valore dell'intervallo nodale. Anche in questo caso, cioè, più il valore d_i è alto più il limite si allontana dal lato i.
Nella didascalia sono riportati i valori degli intervalli nodali d_i, si riferiscono ai lati partendo dal lato evidenziato e proseguendo in senso antiorario.

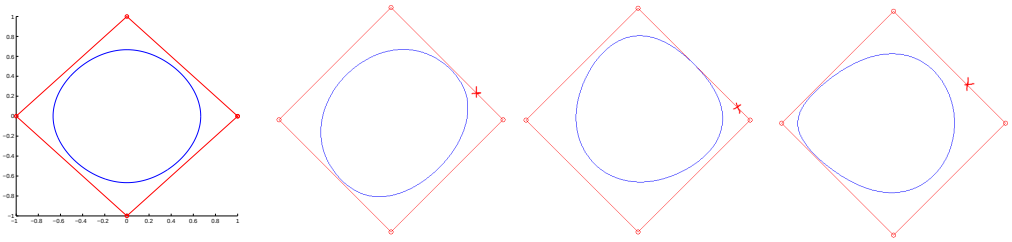


Figura 4.8: Approssimazione non uniforme non stazionaria cubica: B-spline cubica ($d_i \equiv 1$); $d_i = (1,4,1,3)$; $d_i = (1,5,3,4)$ e $d_i = (5,1,1,5)$

4.5. **Interpolazione Uniforme Stazionario**

Gli schemi rappresentanti questa classe sono quelli spiegati negli esempi 1.8, 1.11, 1.14, 1.12 e 1.15; cioè sono tutti quegli schemi chiamati a *n punti*. Nelle seguenti figure sono rappresentati degli esempi con schemi a 4,6,8 e 10 punti.

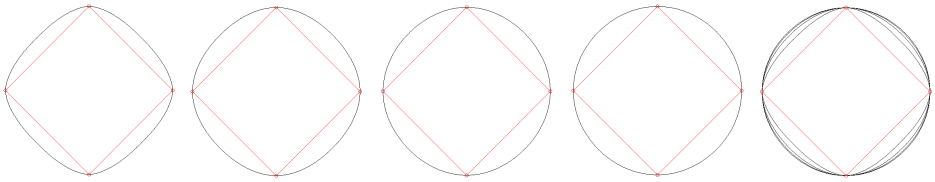


Figura 4.9: Interpolazione uniforme stazionario, poligonale chiusa: 4 Punti, 6 Punti, 8 Punti, 10 Punti e 4,6,8,10 Punti

Si nota che, a parità di punti iniziali, se si utilizzano i metodi a otto o dieci punti, le curve limite calcolate risultano essere più regolari. Lo svantaggio che si ha nel calcolo con metodi a molti punti è che il costo computazionale aumenta. Inoltre, per osservare al meglio le potenzialità di uno schema, c'è bisogno di poligonali iniziali con un numero maggiore o uguale al numero dei punti utilizzati nel metodo. Si può osservare, allora, il seguente esempio di poligonale aperta con 13 punti iniziali.

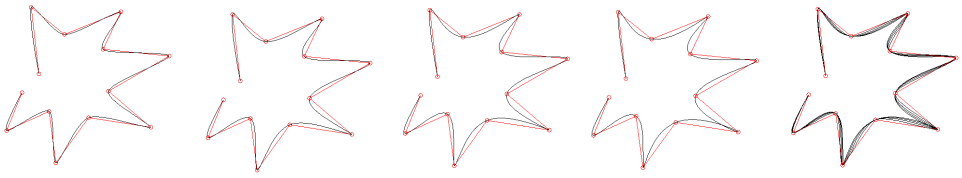


Figura 4.10: Interpolazione uniforme stazionario, poligonale aperta: 4 Punti, 6 Punti, 8 Punti, 10 Punti e 4,6,8,10 Punti

Anche con questo esempio si può osservare che utilizzando più punti la regolarità e la qualità della curva limite sono maggiori con i metodi a otto e dieci punti.

Inoltre, nella teoria presentata nel primo capitolo, ci sono due esempi di schemi che derivano dagli schemi finora visti ma sono modificati dall'introduzione di un parametro ω cioè lo schema a quattro punti ed a sei punti con parametro. Nelle figure seguenti si può osservare il metodo a quattro punti con diversi valori del parametro.

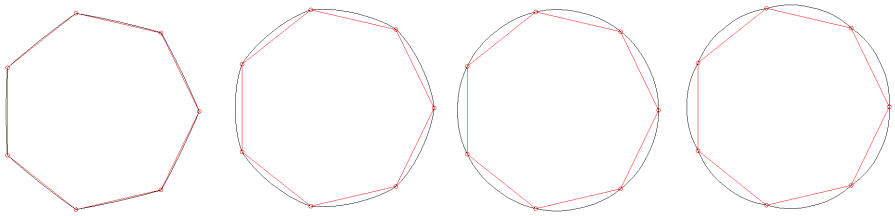


Figura 4.11: Schema a 4 Punti con parametro:$\omega = 0.01$; $\omega = 0.05$; $\omega = 0.07$ e $\omega = 0.08$

Si nota che all'aumentare del parametro la curva limite tende ad essere più lontana dalla poligonale iniziale. Lo stesso accade anche utilizzando il metodo a sei punti con parametro, come si vede nelle seguenti figure.

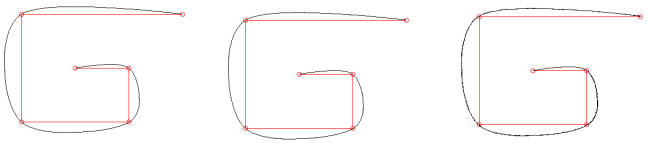


Figura 4.12: Schema a 6 Punti con parametro: $\omega = 0.005$;$\omega = 0.01$ e $\omega = 0.04$

Osservando l'ultima immagine si nota che iniziano ad esserci delle piccole irregolarità nella curva limite; Si osserva infatti che negli esempi precedenti il parametro è sempre compreso nell'intervallo $(0, \frac{1}{8})$; se si prova ad utilizzare un valore diverso si ottengono curve con diverse irregolarità, come si può osservare nelle seguenti immagini.

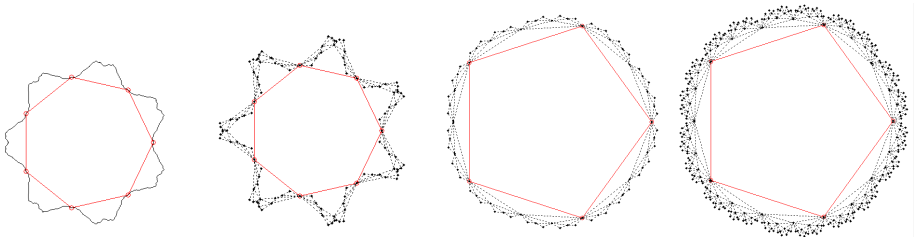


Figura 4.13: Schema a 4 Punti con parametro:$\omega = 0.2$; $\omega = 0.3$; Schema a 6 Punti con parametro $\omega = 0.8$ e $\omega = 1$

4.6. **Interpolazione Uniforme Non Stazionario**
In questa classe ci sono tre esempi di schemi: lo schema interpolante un cerchio, esempio 1.17, lo schema a quattro punti non stazionario, esempio 1.16, e lo schema variazionale, esempio 1.18. Il primo schema è molto particolare perché ha la proprietà di riprodurre un cerchio ma c'è bisogno la poligonale iniziale deve avere i punti equidistanti ed eqis-paziati, oltre a dover essere chiusa. Nelle seguenti figure si hanno le curve finali relative a poligonali iniziali come il pentagono, l'esagono e l'ettagono. Per dimostrare che tale metodo riproduce una circonferenza si riportano anche i grafici delle curvature.

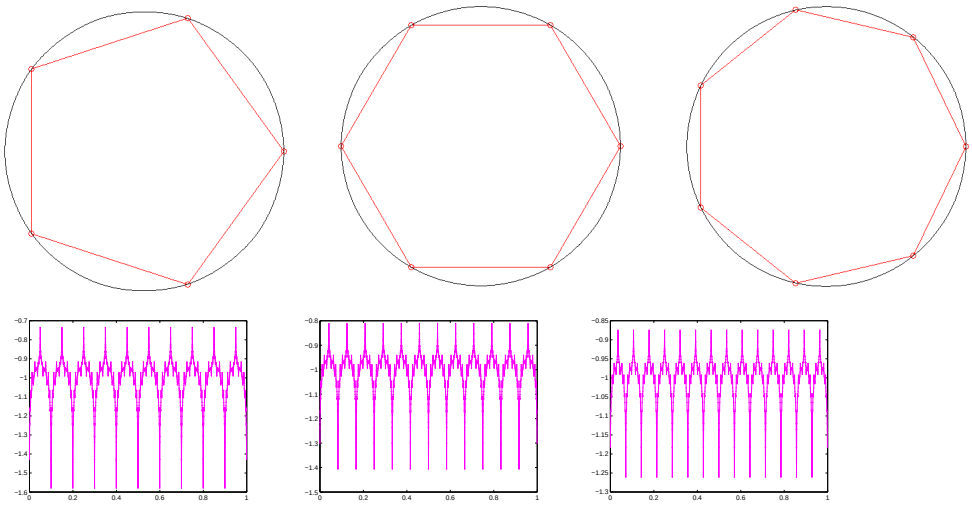


Figura 4.14: Schema interpolante un cerchio: Pentagono; Esagono e Ettagono con relative cuvatura

Per osservare meglio la proprietà di questo schema, si vuole mettere in relazione tale metodo con il quattro punti classico applicati ad un quadrato. Si ottiene quindi

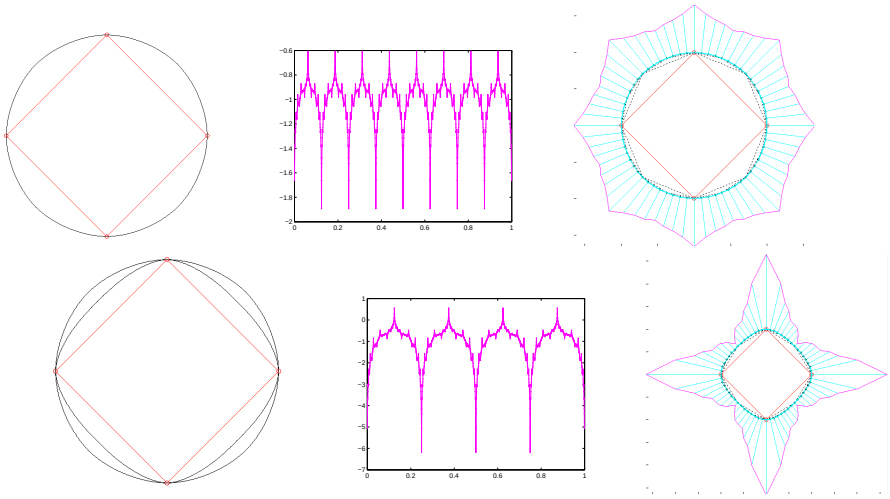


Figura 4.15: Sopra: Schema interpolante un cerchio; curvatura e Curvatura discreta sulla poligonale calcolata al passo $k = 4$. Sotto: Schema a quattro punti classico (interno); curvatura e Curvatura discreta sulla poligonale calcolata al passo $k = 4$

Per quanto riguarda il secondo schema, si ricorda che si può definire il parametro v_0 in modo da avere diverse proprietà della curva limite. Come rappresentato in figura 4.16 si osserva che all'aumentare del valore di v_0 la curva limite tende ad avvicinarsi alla poligonale iniziale.

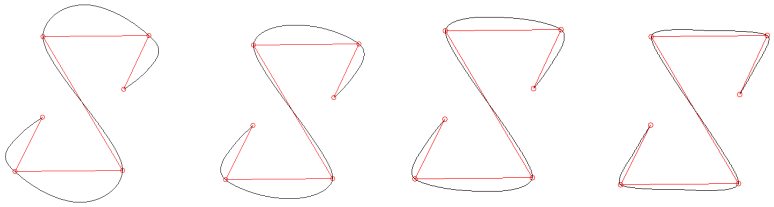


Figura 4.16: Schema Uniforme non Stazionario:$v_0 = -0.5$; $v_0 = 0$; $v_0 = 1$ e $v_0 = 5$

Si vuole ora osservare l'effetto dell'applicazione del metodo su un poligono regolare con i punti equidistanti inscritto nel cerchio unitario. In questo caso si prende in considerazione il quadrato e, nelle immagini della figura 4.17, si è calcolato il cerchio utilizzando il metodo a quattro punti interpolante un cerchio descritto nel paragrafo precedente.

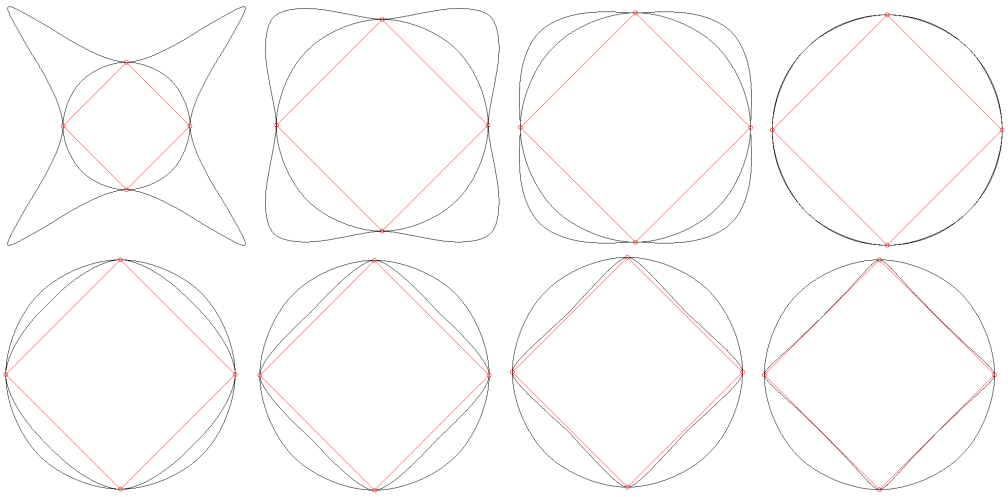


Figura 4.17: Schema Uniforme non Stazionario:$v_0 = -0.95$; $v_0 = -0.75$; $v_0 = -0.5$ $v_0 = 0$; $v_0 = 1$; $v_0 = 5$; $v_0 = 25$ e $v_0 = 500$

Si nota che scegliendo il parametro $v_0 < 1$ la poligonle risulta essere definita all'esterno della circonferenza unitaria si allontana sempre di più dalla poligonale iniziale; se, invece, $v_0 \geq 1$ la curva limite è definita all'interno della circonferenza unitaria e tende alla poligonale iniziale all'aumentare del parametro. Un fatto molto interessante da osservare è che se $v_0 = cos(\frac{2\pi}{N})$ con N il numero dei punti iniziali, in questo caso 4, si ha che la curva limite è esattamente il cerchio unitario.

Il terzo schema è quello variazionale presentato da Kobbelt. In fase di crazione del programma si è implementato solo per poligonali chiuse. Lo schema produce delle curve molto regolari e di alta qualità perché si sfrutta l'approccio variazionale. Nell'esempio in figura 4.19 si mette in relazione il metodo con il classico metodo a quattro punti.

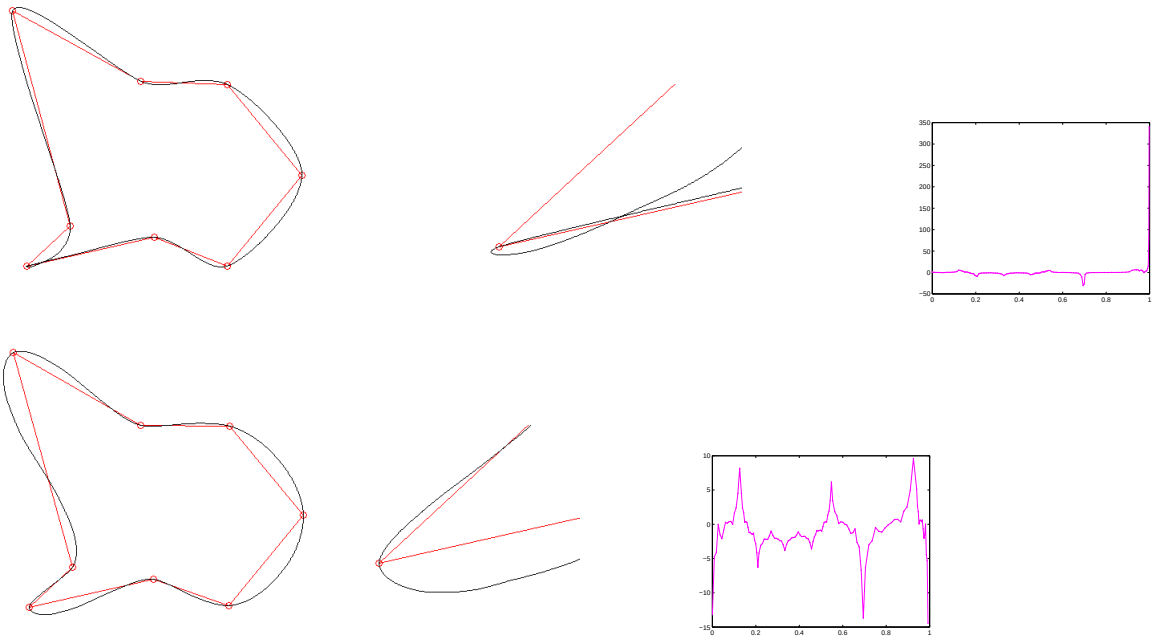


Figura 4.18: Sopra: schema a quattro punti, zoom-in nel punto di irregolare e curvatura. Sotto: Schema Variazionale, zoom-in nel punto di irregolare e curvatura.

4.7. **Interpolazione Non Uniforme Stazionario**

Lo schema NULI dell'esempio 1.19 rappresenta questa particolare classe di schemi. Per definire tale metodo c'è bisogno di fissare i parametri λ, uno per ogni lato della poligonale. Si ricorda che tali valori devono essere nell'intervallo $[0, 1]$ e che a seconda che λ sia in $(0, \frac{1}{2})$ o $(\frac{1}{2}, 1)$ si hanno diversi coefficienti che definiscono il metodo. Si forniscono degli esempi con diversi valori di questi parametri in figura 4.19.

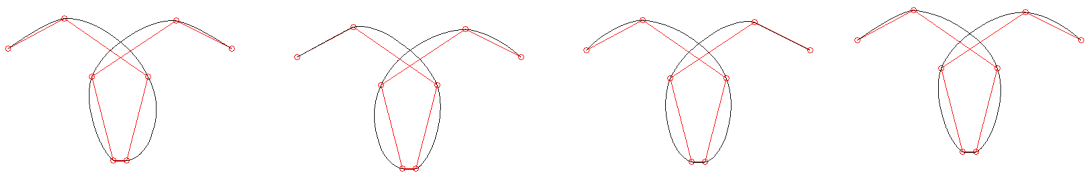


Figura 4.19: Metodo NULI: $\lambda = \{0.5, 0.5, 0.1, 0.1, 0.1, 0.5, 0.5\}$; $\lambda = \{0.9, 0.8, 0.7, 0.1, 0.7, 0.8, 0.9\}$; $\lambda = \{0.01, 0.2, 0.5, 0.5, 0.5, 0.2, 0.01\}$ e $\lambda = \{0.5, 0.01, 0.7, 0.4, 0.2, 0.5, 0.9\}$

Si nota che a seconda del valore dei coefficienti si può variare la convessità della curva; se si provano ad utilizzare dei metodi uniformi per la stessa poiligonale iniziale si hanno diversi

risultati. Come mostrato in figura 4.20, si può osservare, in particolare, l'incongruenza data dal metodo non stazionario uniforme con $v_0 = -0.1$

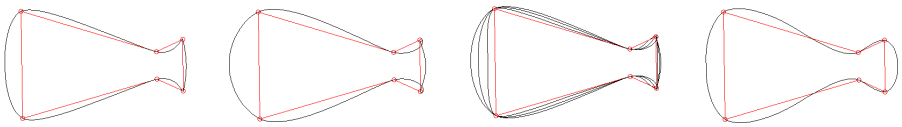


Figura 4.20: schema a quattro punti classico; schema non stazionario uniforme con $v_0 = -0.1$; schema non stazionario uniforme con $v_0 = 0.2, 1$ e 5 e schema variazionale

4.8. **Interpolazione Non Uniforme Non Stazionario**

Il rappresentante di questa classe è lo schema dell'esempio 1.20. Tale schema è una variante dello schema a quattro punti con parametro, in cui non c'è un unico valore ma ci sono tanti valori del parametro diversi quanti sono i vertici della poligonale iniziale. In figura 4.21 sono riportati alcuni esempi.

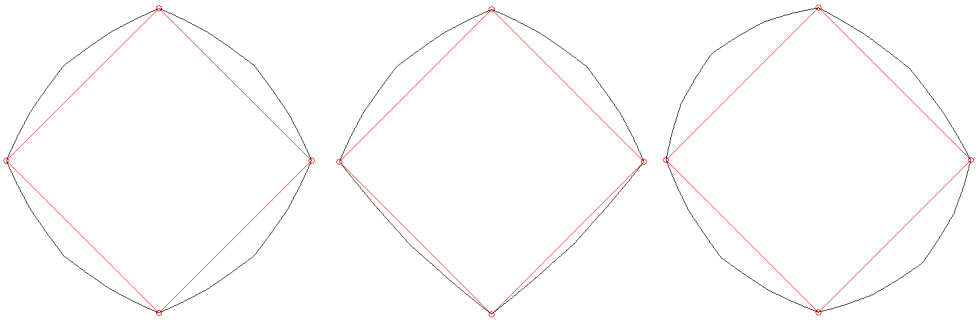


Figura 4.21: Metodo Non Uniforme Non Stazionario: $\omega_i \equiv 0.0625$; $\omega = \{0.0625, 0.0625, 0.02, 0.02\}$; $\omega = \{0.05, 0.1, 0.07, 0.09\}$, dove si parte dal vertice a destra e si procede in senso antiorario

Si nota che a seconda che il valore del vertice sia più o meno grande la curva limite tenderà ad essere più vicina al vertice considerato.
Si osserva, poi, che i parametri utilizzati sono tutti compresi nell'intervallo $(0, \frac{1}{8})$; se si prova con altri valori i risultati cambiano molto e si hanno delle curve non regolari, come mostrato in figura 4.22.

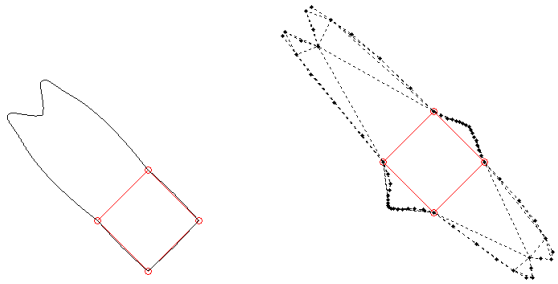


Figura 4.22: Metodo Non Uniforme Non Stazionario: $\omega = \{0.01, 0.8, 0.01, 0.0.1\}$ e $\omega = \{0.1, 0.9, 0.2, 0.7\}$ quattro passi, dove si parte dal vertice a destra e si procede in senso antiorario

4.9. **Interpolazione Non Lineare**

Lo schema non lineare studiato è il metodo del controllo della tangente, per il quale c'è bisogno di stabilire il valore del parametro ω. In figura 4.23 sono presenti alcuni esempi. Si nota che più grande risulta essereil parametro più la curva limite si allontana dalla

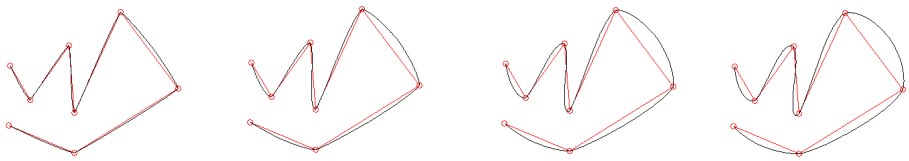


Figura 4.23: Metodo Non Lineare: $\omega = 0.02$; $\omega = 0.05$; $\omega = 0.08$ e $\omega = 0.1$

poligonale iniziale.
Si ha, inoltre, che lo schema riesce ad eliminare o a migliorare delle piccole incongruenze che sono presenti nello schema a quattro punti classico come si può meglio osservare nella figura 4.24.

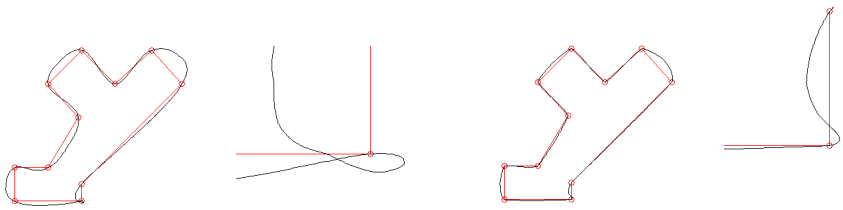


Figura 4.24: Metodo a quattro punti con $\omega = \frac{1}{12}$ e ingrandimento del dettaglio; Metodo non lineare con $\omega = 0.05$ con ingandimento del dettaglio

4.2 Scaling

Per la sperimentazione dello scaling si prendono in considerazione due immagini digitali, una a livelli di grigio ed una a colori, e si provano i vari metodi implementati. Nel caso dell'immagine .tif, nella figura 4.2 si ha l'immagine originale e la porzione di essa scelta per lo scaling.

Figura 4.25: Immagine originale e porzione

Si presentano ora, nelle seguenti figure, i risultati della scaling x2, x4 e x8 relativi alla porzione di immagine scelta, ottenuti utilizzando i vari metodi. In particolare si utilizzano i metodi B-spline lineare, quadratico e cubico; il quattro punti con parametro $\omega = 0.05$ e $\omega = 0.1$; i metodi a N punti con N=4,6,8 e 10; e il metodo non lineare con $\omega = 0.05$ e $\omega = 0.0625$. Di seguito per ogni metodo ci sono tre immagini: la prima rappresenta lo scaling x2, la seconda x4 e la terza l'ingrandimento x8.

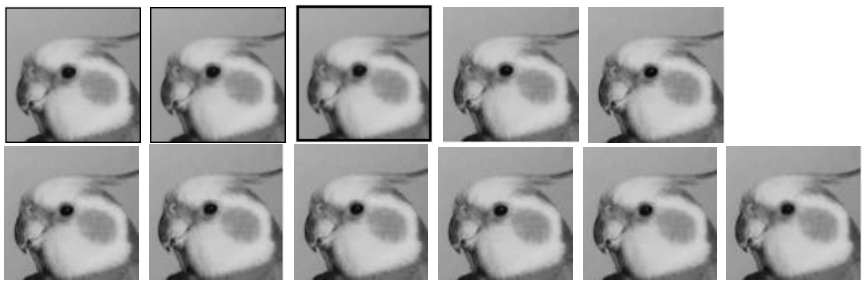


Figura 4.26: Prima riga: schema B-spline lineare, quadratico e cubico; quattro punti con $\omega = 0.05$ e $\omega = 0.1$. Seconda riga: schema a quattro, sei, otto, dieci punti; non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

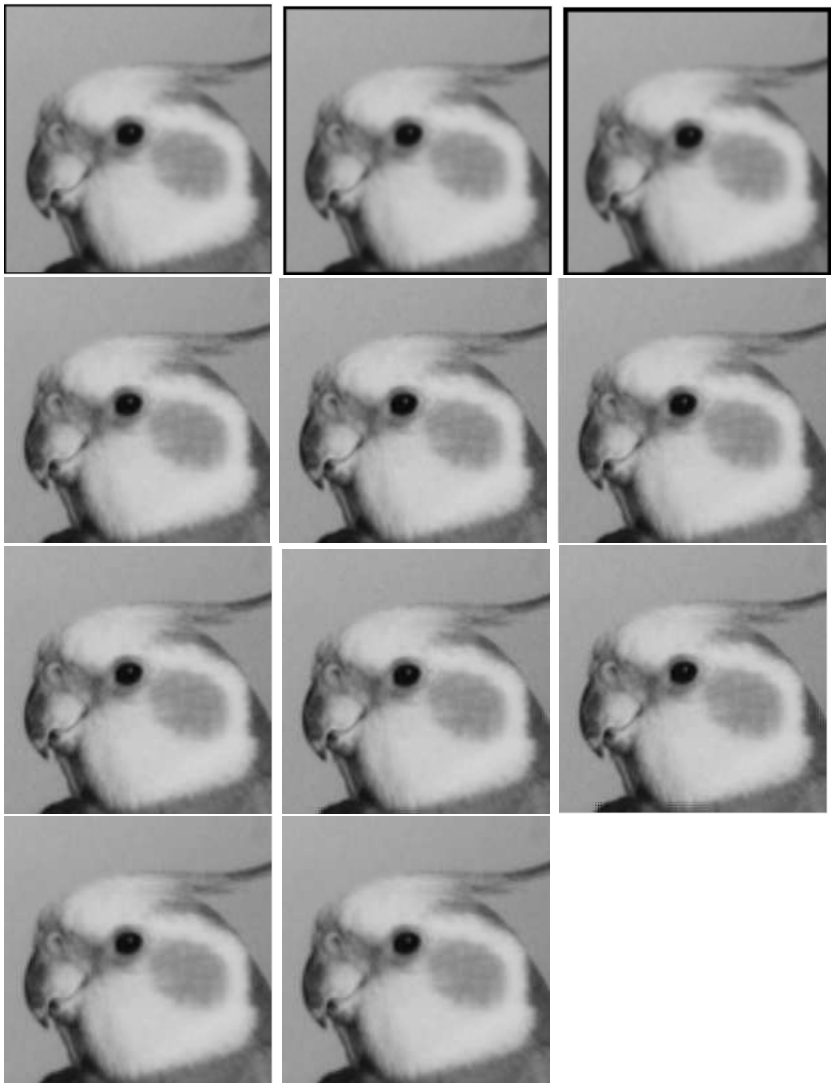


Figura 4.27: Prima riga: schema B-spline lineare, quadratico e cubico; Seconda riga: quattro punti con $\omega = 0.05$; $\omega = 0.1$ e $\omega = 0.0625$; Terza riga: schema a sei, otto, dieci punti; Quarta riga: schema non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

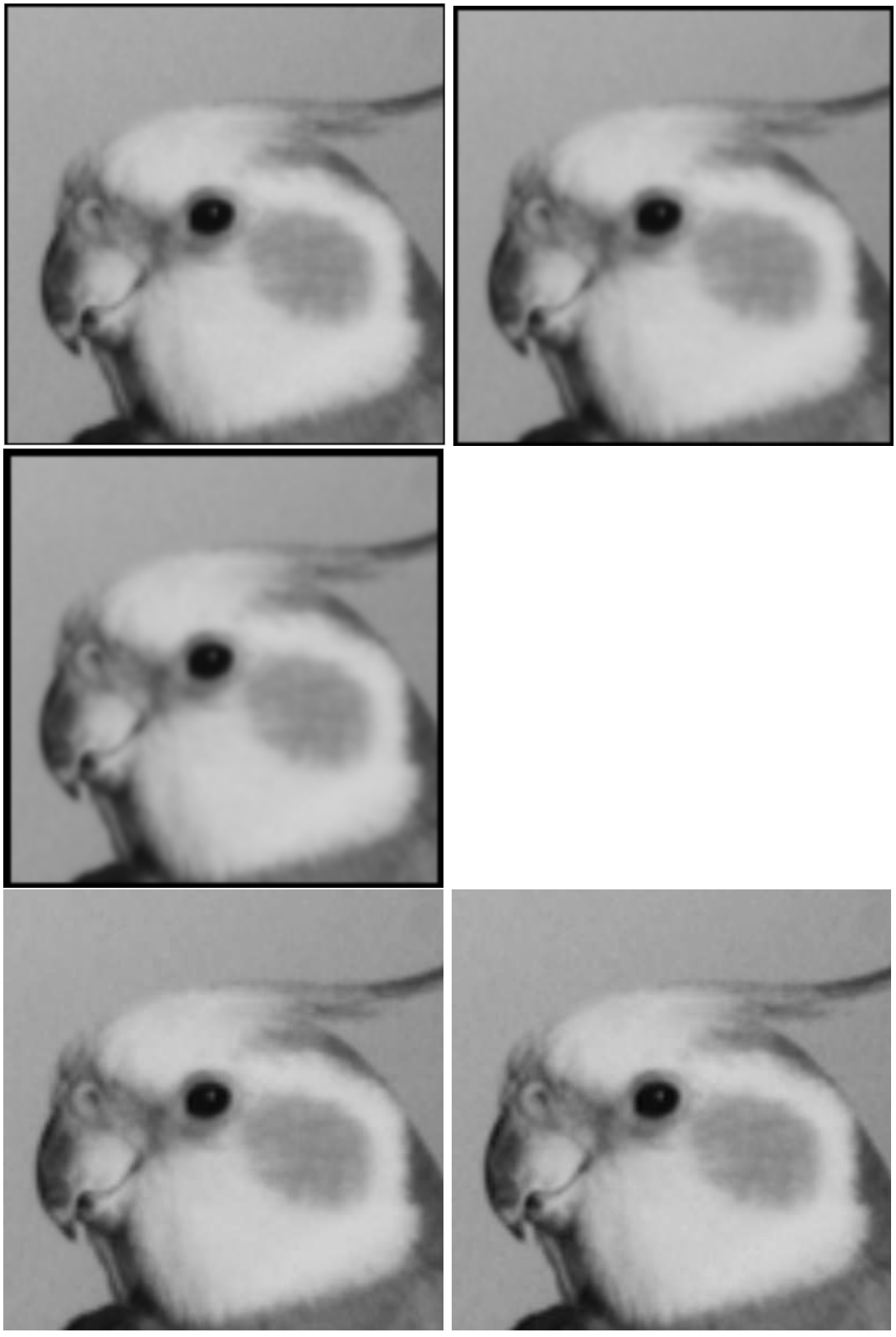


Figura 4.28: schema B-spline lineare, quadratico e cubico; quattro punti con $\omega = 0.05$ e $\omega = 0.1$.

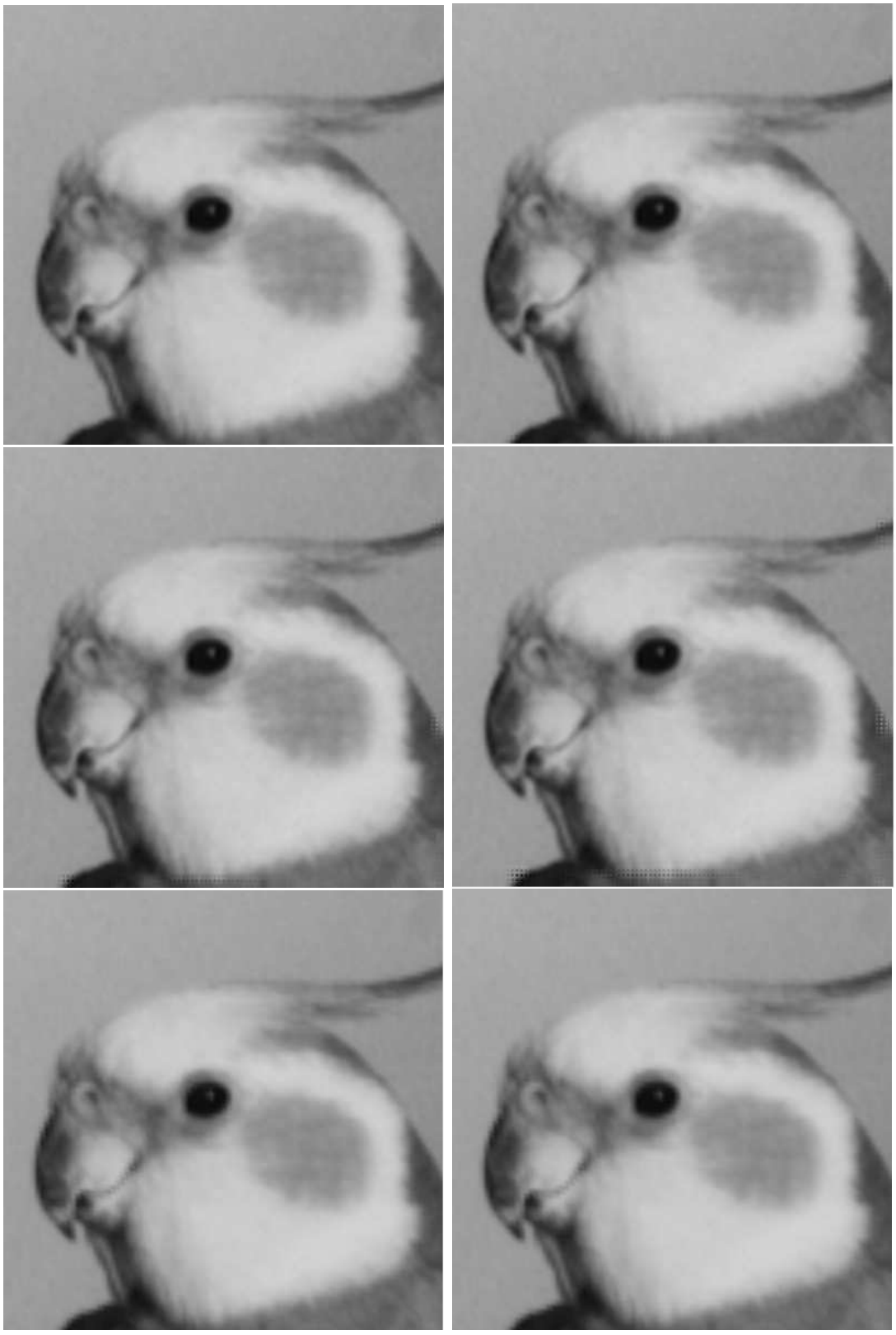


Figura 4.29: schema a quattro, sei, otto, dieci punti; non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

Si osserva che lo scaling in cui le dimensioni raddoppiano risulta sempre essere corretto e non ci sono evidenti differenze nei metodi scelti. Lo stesso discorso vale per lo scaling x4; mentre si iniziano a notare delle differenze osservando le immagini ingrandite otto volte. In questo caso le immagini risultanti dagli schemi approssimanti tendono ad essere un pò sfocate mentre, per gli schemi interpolanti, in linea generale, tendono ad essere meglio definite. In questo particolare esempio si ha che se si utilizza il metodo a quattro punti con $\omega = 0.1$ l'immagine sembra essere meglio definita rispetto all'utilizzo di $\omega = 0.05$. Poi se si calcolano i risultati con gli schemi a N punti si nota che maggior numero punti si utilizzano e maggiore risulta essere la qualità dell'immagine. Infine, osservando i risultati del metodo non lineare, si ha un'inversione di tendenza: per quanto finora detto dovrebbero essere ben definiti in quanto schemi interpolanti ma risultano molto più sfocati qualunque sia il valore del parametro ω utilizzato.

Osservando anche i risultati dell'immagine a colori si nota che rimangono valide le considerazioni appena enunciate. Per effettuare lo scaling si è utilizza l'immagine in figura 4.2.

Figura 4.30: Immagine originale e porzione

Di seguito sono riportati le immagini ottenute; anche in questo caso la prima immagine di ogni set di tre rappresenta lo scaling x2, la seconda x4 e la terza x8.

Figura 4.31: Prima riga: schema B-spline lineare, quadratico e cubico; quattro punti con $\omega = 0.05$ e $\omega = 0.1$. Seconda riga: schema a quattro, sei, otto, dieci punti; non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

Anche in questo caso lo scaling x2 e x4 risulta produrre delle immagini di altà qualità e non sembrano esserci differenze visibili tra i vari schemi. Più interessante è osservare i risultati prodotti dallo scaling x8. Si nota che gli schemi approssimanti e lo schema non lineare producono delle immagini molto sfocate, mentre gli altri metodi interpolanti forniscono delle immagini meglio definite. Anche in questo caso gli schemi che utilizzano un numero alto di punti producono delle immagini migliori.

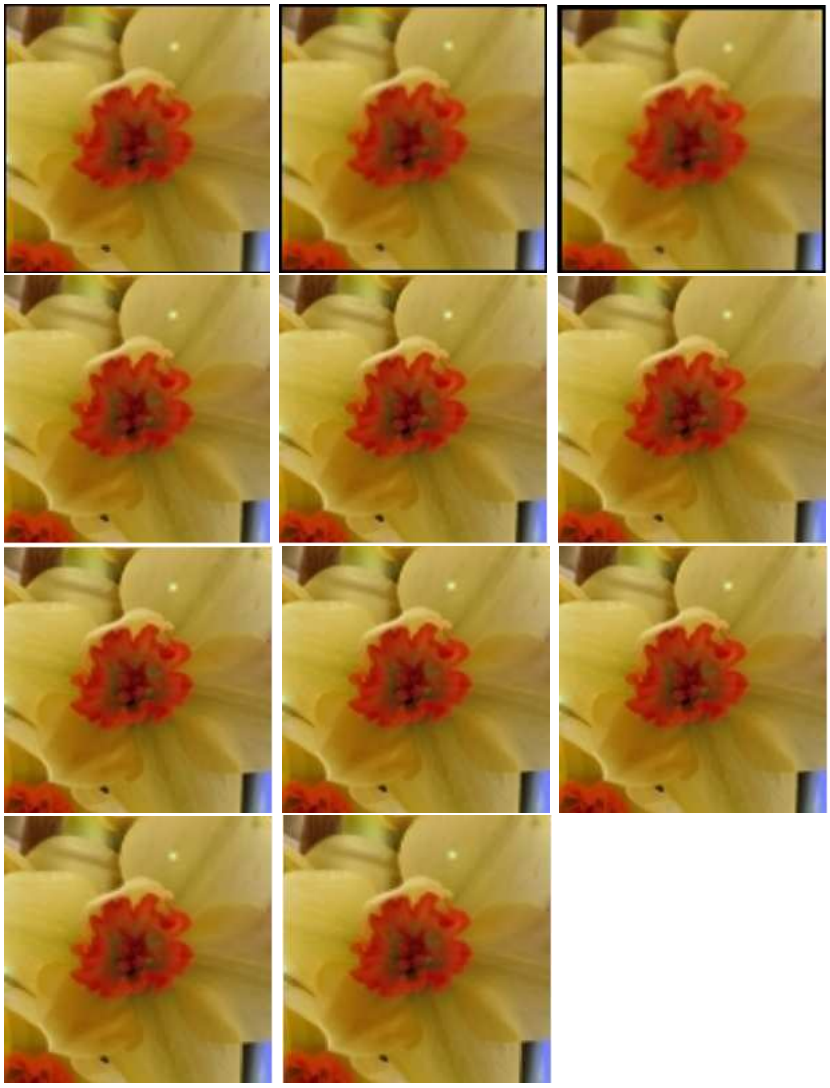


Figura 4.32: Prima riga: schema B-spline lineare, quadratico e cubico; Seconda riga: quattro punti con $\omega = 0.05$; $\omega = 0.1$ e $\omega = 0.0625$; Terza riga: schema a sei, otto, dieci punti; Quarta riga: schema non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

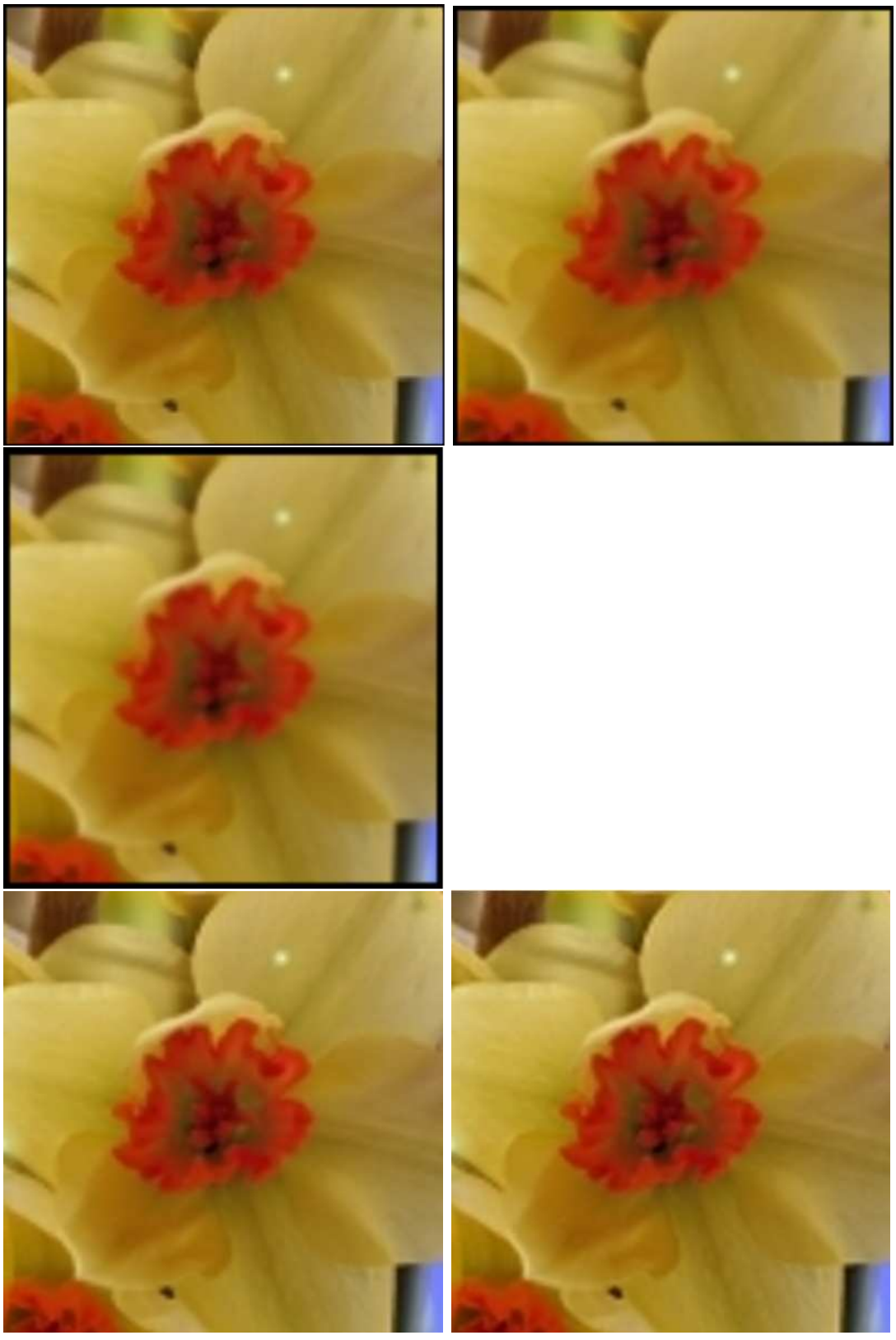


Figura 4.33: schema B-spline lineare, quadratico e cubico; quattro punti con $\omega = 0.05$ e $\omega = 0.1$.

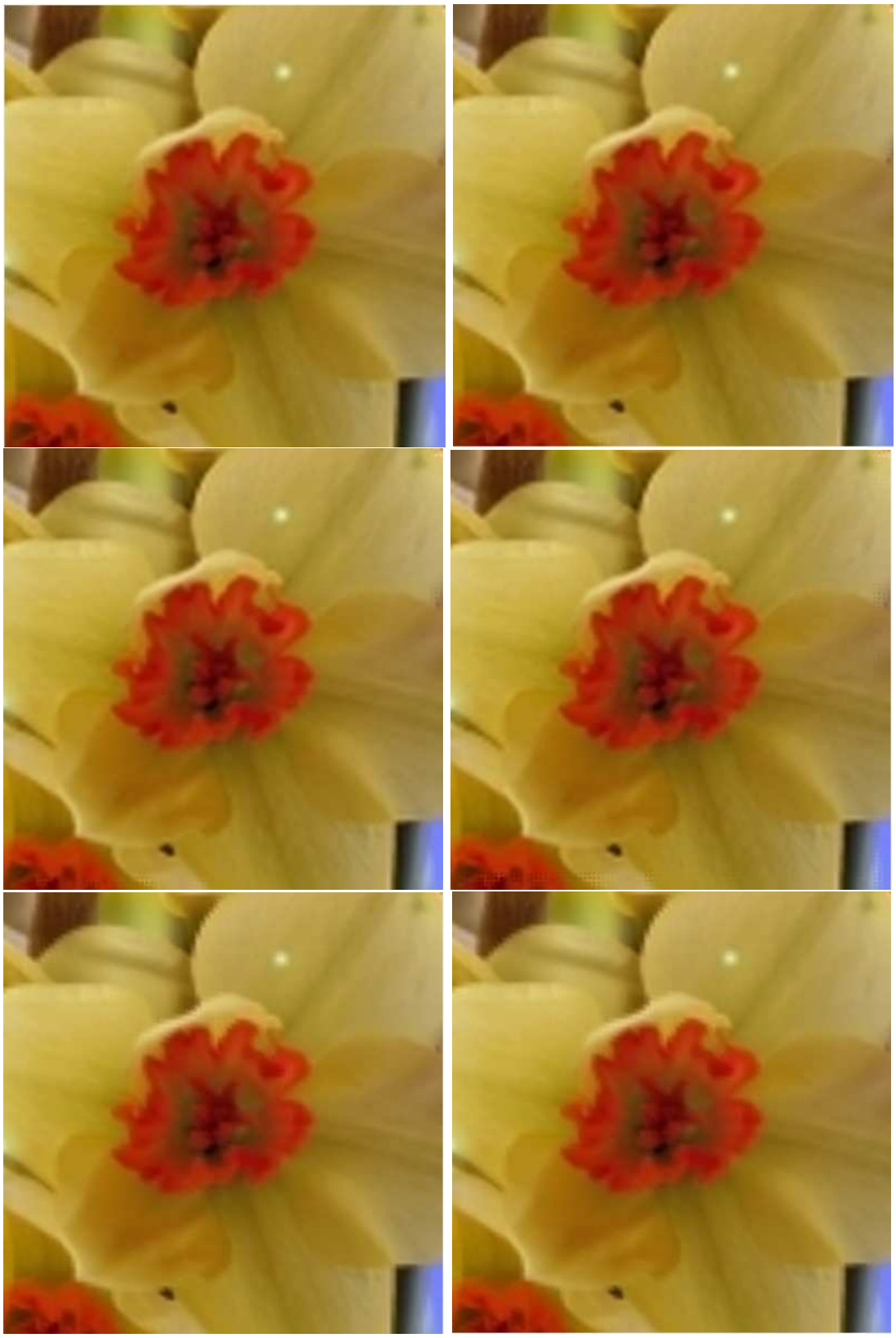


Figura 4.34: schema a quattro, sei, otto, dieci punti; non lineare con $\omega = 0.0625$ e $\omega = 0.05$.

4.3 Compressione

Nel secondo capitolo si è parlato di compressione di segnali e di immagini digitali nel senso che, invece di salvare il segnale o l'immagine, si salvano solo dei punti iniziali e l'errore calcolato come differenza tra il segnale (l'immagine) originale e il segnale (l'immagine) calcolato. Per ottenere la compressione vera e propria c'è bisogno di salvare meno dati possibili utilizzando quindi meno memoria. Tutti i programmi implementati sono scritti con il codice Matlab, quindi si è pensato di salvare il vettore o la matrice che rappresentano le differenze relative al segnale o all'immagine in formato `sparse`. Questo particolare tipo di formato non tiene conto degli zeri di una matrice, cioè salva solo gli elementi diversi da zero. È necessario allora capire come vengono salvati i valori di una matrice in memoria. Ogni numero si rappresenta con 8 bytes e si memorizza come *double*, cioè il formato a doppia precisione. Sia M una matrice mxn allora:

- se si salva con il formato usuale in memoria verrà allocata tutta la matrice in memoria e si occuperanno 8xmxn bytes; mxn il numero di elementi da allocare e 8 bytes per ogni elemento.

- se si memorizza in fomato sparse in memoria verranno allocati 3 array differenti: A contenente gli elementi non nulli; IA contenente gli indici riferiti all'array A del primo elemento non nullo di ogni riga, cioè $IA(i)$ è l'indice in A del primo elemento non nullo della matrice M; e JA contiene l'indice delle colonne in M di ogni elemento di A, cioè $JA(i)$ è l'indice della colonna in cui si trova il i-esimo elemento di A. Sia nnz il numero degli elementi non nulli nella matrice M allora: A e JA sono vettori di lunghezza nz e IA è un vettore di lunghezza $m+1$, m per il numero delle righe più un elemento dato da nnz. Quindi in memoria si occuperanno 8xnz^2+8x$(m+1)$. Si nota che per utilizzare le proprietà delle matrici in formato sparse occorre che ci sia un elevato numero di zeri in particolare deve valere la seguente disuguaglianza:

$$nnz < \frac{m(n-1)-1}{2}$$

Esempio 4.1. Se si considera la matrice M di dimensioni 3x4:

$$M = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 3 & 9 & 0 \\ 0 & 1 & 4 & 0 \end{pmatrix}$$

si ha che utilizzando il formato usuale si occupano 3x4x8=96 bytes. Se si utilizza il formato sparse si ha che $nz = 6$ e gli array salvati sono:

$$A = \begin{pmatrix} 1 & 2 & 3 & 9 & 1 & 4 \end{pmatrix}$$
$$IA = \begin{pmatrix} 0 & 2 & 4 & 6 \end{pmatrix}$$
$$JA = \begin{pmatrix} 0 & 1 & 1 & 2 & 1 & 2 \end{pmatrix}$$

si osserva che $nnz = 6 > \frac{m(n-1)-1}{2} = 4$ quindi non sarà conveniente sfruttare questo formato.
Si ha infatti che:

$$\begin{aligned} A \rightarrow &\quad 6\text{x}8 = 48 \text{ bytes} \\ IA \rightarrow &\quad (3+1)\text{x}8 = 24 \text{ bytes} \\ JA \rightarrow &\quad 6\text{x}8 = 48 \text{ bytes} \end{aligned}$$

per un totale di 120 bytes.
Se invece

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

si ha che utilizzando il formato usuale si occupano sempre 3x4x8=96 bytes. Se si utilizza il formato sparse si ha che $nz = 9$ e gli array salvati sono:

$$\begin{aligned} A &= \begin{pmatrix} 1 & 9 & 1 \end{pmatrix} \\ IA &= \begin{pmatrix} 0 & 1 & 2 & 3 \end{pmatrix} \\ JA &= \begin{pmatrix} 0 & 1 & 2 \end{pmatrix} \end{aligned}$$

si osserva che questa volta $nnz = 3 < \frac{m(n-1)-1}{2} = 4$ quindi non sarà conveniente sfruttare questo formato.
Si ha infatti che:

$$\begin{aligned} A \rightarrow &\quad 3\text{x}8 = 24 \text{ bytes} \\ IA \rightarrow &\quad (3+1)\text{x}8 = 24 \text{ bytes} \\ JA \rightarrow &\quad 3\text{x}8 = 24 \text{ bytes} \end{aligned}$$

per un totale di 72 bytes.

Di seguito sono riportati degli esempi in cui sono evidenziati il numero degli zeri e lo spazio occupato in memoria.

4.3.1 Compressione di Segnali

Si vuole effettuare la compressione di segnali utilizzando la discretizzazione di due funzioni continue quali il seno e il coseno. Si parte da un segnale discreto composto da 257 valori, cioè $2^8 + 1$. Si effettua la compressione, utilizzando 9 punti iniziali, con i metodi a due, quattro, sei e otto punti e fissando diverse tolleranze. Nelle seguenti tabelle sono riportati il numero di zeri, e la percentuale degli stessi, in ciascun vettore delle differenze e il numero di bytes necessari per salvare questo vettore nel caso sparse. Nella lettura della tabella bisogna tener conto che la memoria occupata per salvare il vettore nel formato usuale è di 2,056 bytes (257 x 8)

Tol	n	k	%zeri	sparse
0.01	2	41	15.95	3,472
	4	201	78.21	*912*
	6	201	78.21	*912*
	8	165	64.20	*1,488*
0.001	2	9	3.50	3,984
	4	43	16.73	3,440
	6	115	44.75	2,288
	8	95	36.96	2,608
0.0001	2	9	3.50	3,984
	4	13	5.06	3,920
	6	29	11.28	3,664
	8	57	22.18	3,216
0.00001	2	9	3.50	3,984
	4	9	3.50	3,984
	6	11	4.28	3,952
	8	11	4.28	3,952

Tol	n	k	%zeri	sparse
0.01	2	39	15.18	3,504
	4	257	100.00	*32*
	6	166	64.59	*1,472*
	8	163	63.42	*1,520*
0.001	2	9	3.50	3,984
	4	57	22.18	3,216
	6	101	39.30	2,512
	8	97	37.74	2,576
0.0001	2	9	3.50	3,984
	4	17	6.61	3,856
	6	26	10.12	3,712
	8	47	18.29	3,376
0.00001	2	9	3.50	3,984
	4	9	3.50	3,984
	6	11	4.28	3,952
	8	13	5.06	3,920

Tabella 4.1: Seno e Coseno 256pt Tol= tolleranza, n= schema a n punti, k=numero di zeri, %zeri=percentuale del numero degli zeri e sparse= numero di bytes necessari per salvare il vettore in formato sparse

Prima di commettare i risultati ottenuti è bene precisare che uno schema si considera migliore se produce un vettore delle differenze con un alto numero di zeri, perché in questo modo si può salvare in formato sparse, ad esempio, e si raggiunge l'obbiettivo della compressione, cioè la riduzione di memoria utilizzata.
Ora, osservando le tabelle, si ha che il metodo del punto medio o a due punti non comviene mai; solo con una tolleranza alta, 0.01, si ah che il numero di zeri non corrisponde solo al numero dei punti iniziali. Considerando invece gli altri schemi si ha che per la tolleranza 0.01 lo schema migliore è quello a quattro punti, per tolleranza 0.001 comprime meglio lo schema a sei punti; invece abbassando ulteriormente la tolleranza

risulta migliore il metodo a otto punti. Si può quindi affermare, in via generale, che abbassando la tolleranza risultano migliori gli schemi a molti punti.
La stessa tendenza è già stata vista da Floater in [23].
Spostando ora l'attenzione sui risultati ottenuti con l'utilizzo dei vettori in formato sparse si vede che non conviene quasi mai utilizzare questo formato. I vettori sparse convengono solo nei valori evidenziati e sono tutti con tolleranza alta a 0.01.
Si può osservare anche il risultato degli stessi procedimenti applicati sempre al seno e al coseno, con 9 punti iniziali, ma aumentanto i valori della discretizzazione a 1025 (2^10-1); in questo caso il vettore delle differenze nel formato usuale utilizza 8,200 bytes per essere salvato.

Tol	n	k	%zeri	sparse
0.01	2	145	14.15	14,096
	4	801	78.15	*3,600*
	6	799	77.95	*3,632*
	8	659	64.29	*5,872*
0.001	2	13	1.27	16,208
	4	175	17.07	13,616
	6	453	44.20	9,168
	8	369	36.00	10,512
0.0001	2	9	0.88	16,272
	4	41	4.00	15,760
	6	107	10.44	14,704
	8	207	20.20	13,104
0.00001	2	9	0.88	16,272
	4	11	1.07	16,240
	6	27	2.63	15,984
	8	21	2.05	16,080

Tol	n	k	%zeri	sparse
0.01	2	138	13.46	14,208
	4	1025	100.00	*32*
	6	659	64.29	*5,872*
	8	643	62.73	*6,128*
0.001	2	13	1.27	16,208
	4	233	22.73	12,688
	6	394	38.44	10,112
	8	373	36.39	10,448
0.0001	2	9	0.88	16,272
	4	61	5.95	15,440
	6	87	8.49	15,024
	8	171	16.68	13,680
0.00001	2	9	0.88	16,272
	4	13	1.27	16,208
	6	21	2.05	16,080
	8	31	3.02	15,920

Tabella 4.2: Seno e Coseno 1024pt: Tol= tolleranza, n= schema a n punti, k=numero di zeri, %zeri=percentuale del numero degli zeri e sparse= numero di bytes necessari per salvare il vettore in formato sparse

Anche in questo caso valgono le osservazioni dell'esempio precedente. Se si considerano il numero degli zeri prodotti da ciascun metodo si ha che per la tolleranza fissata a 0.01 lo schema migliore risulta quello a quattro punti; per la tolleranza a 0.001 conviene lo schema a sei punti e abbassando ulteriormente il migliore risulta essere il metodo a otto punti.
Osservando meglio le percentuali di zeri in entrambi gli esempi, cioè utilizzando la discretizzazione a 257 0 1025 punti, si ha che utilizzando un numero di punti più basso i risultati migliorano in quanto tale percentuale risulta essere maggiore.

Considerando, invece, lo spazio utilizzato in memoria si ha che, anche in questo esempio, conviene utilizzare il formato sparse solo se la tolleranza è di 0.01.

4.3.2 Compressione di Immagini

In questa ultima sezione si hanno degli esempi di compressione di immagini digitali. Si è considerato sia un'immagine .tif che un'immagine .jpg. Per effettuare la compressione si sono utilizzati i metodi a due, quattro, sei e otto punti; il metodo a quattro punti con il parametro $\omega = 0.1$ e $\omega = 0.05$; e il metodo non lineare con $\omega = 0.0625$ e $\omega = 0.05$. Per ogni schema si riportano le prove effettuate con diverse tolleranze. Ci sono le immagini delle ricostruzioni e ci sono diverse tabelle. Nella prima tabella vengono riportati il numero degli zeri della matrice delle differenze; la percentuale degli zeri rispetto al totale degli elementi e il numero di bytes utilizzati nel caso la matrice venga salvata in formato sparse. Nella seconda tabella, invece, ci sono tutte le informazioni riguardanti la qualità dell'immagine ricostruita divise per tolleranza.
Come primo esempio si comprime l'immagine in figura 4.35, di dimensione 257 x 257 pixels.

Figura 4.35: Immagine Originale

Nella tabella 4.3 sono riportati gli zeri, la percentuale degli zeri e il numero di bytes necessari per salvare la matrice delle differenze in formato sparse; si ricorda che tale matrice ha dimensione 257 x 257 quindi salvata nel formato usuale occupa 528,392 bytes. Inoltre si ha che per la ricostruzione si sono utilizzati 9^2 punti iniziali, cioè inizialmente si applicano i procedimenti di suddivisione a matrici 36 x 36, per poi iterare il procedimento arrivando a matrici 257 x 257; in particolare sono necessari quattro passi di suddivisione.

Schema a 2 punti			
Tol	# Zeri	%Zeri	Bytes
0.25	56741	**85.91**	*150,992*
0.18	52723	79.82	*215,280*
0.15	50397	76.30	*252,496*
0.10	44979	68.10	*339,184*
0.08	41707	63.15	*391,536*
0.05	35481	**53.72**	*491,152*
0.01	16579	**25.10**	793,584
0.001	2087	**3.16**	1,025,456

Schema a 4 punti			
Tol	# Zeri	%Zeri	Bytes
0.25	56343	85.30	*157,360*
0.18	52607	79.65	*217,136*
0.15	50637	76.67	*248,656*
0.10	45815	**69.37**	*325,808*
0.08	42543	**64.41**	*378,160*
0.05	34787	52.67	*502,256*
0.01	13958	21.13	835,520
0.001	1635	2.48	1,032,688

Schema a 4 pt w=0.1			
Tol	# Zeri	%Zeri	Bytes
0.25	55682	84.30	*167,936*
0.18	51896	78.57	*228,512*
0.15	49401	74.79	*268,432*
0.10	41742	63.20	*390,976*
0.08	37258	56.41	*462,720*
0.05	28512	43.17	602,656
0.01	8039	12.17	930,224
0.001	844	1.28	1,045,344

Schema a 4 pt w=0.05			
Tol	# Zeri	%Zeri	Bytes
0.25	55749	84.41	*166,864*
0.18	51852	78.51	*229,216*
0.15	49527	74.99	*266,416*
0.10	44036	66.67	*354,272*
0.08	40763	61.72	*406,640*
0.05	32722	49.54	535,296
0.01	10620	16.08	888,928
0.001	1137	1.72	1,040,656

Schema a 6 punti			
Tol	# Zeri	%Zeri	Bytes
0.25	56340	85.30	*157,408*
0.18	52732	79.84	*215,136*
0.15	50717	**76.79**	*247,376*
0.10	45214	68.46	*335,424*
0.08	41385	62.66	*396,688*
0.05	33478	50.69	*523,200*
0.01	12234	18.52	863,104
0.001	1404	2.13	1,036,384

Schema a 8 punti			
Tol	# Zeri	%Zeri	Bytes
0.25	56575	85.66	*153,648*
0.18	52885	**80.07**	*212,688*
0.15	50626	76.65	*248,832*
0.10	44229	66.96	*351,184*
0.08	40336	61.07	*413,472*
0.05	32228	48.79	543,200
0.01	10905	16.51	884,368
0.001	1269	1.92	1,038,544

non lineare w=0.0625			
Tol	# Zeri	%Zeri	Bytes
0.25	55753	84.41	*166,800*
0.18	50511	76.48	*250,672*
0.15	47541	71.98	*298,192*
0.10	41163	62.32	*400,240*
0.08	38151	57.76	*448,432*
0.05	31894	48.29	548,544
0.01	12735	19.28	855,088
0.001	1477	2.24	1,035,216

non lineare w=0.05			
Tol	# Zeri	%Zeri	Bytes
0.25	55734	84.38	*167,104*
0.18	50518	76.49	*250,560*
0.15	47525	71.95	*298,448*
0.10	41197	62.37	*399,696*
0.08	38167	57.79	*448,176*
0.05	31919	48.33	548,144
0.01	12870	19.49	852,928
0.001	1499	2.27	1,034,864

Tabella 4.3: Tabella relativa all'immagine 4.35

In tabella 4.3 sono evidenziati in grassetto i migliori ed in grigio i peggiori risultati per ciascuna tolleranza. Si nota che per le tolleranze 0.25, 0.05, 0.01 e 0.001 risulta essere migliore il metodo a due punti. Il metodo a quattro punti classico è il migliore se la tolleranza è 0.1 o 0.08; invece per le altre tolleranze considerate, cioè 0.15 e 0.18, risultano convenienti i metodi a sei punti e a otto punti rispettivamente.
Osservando, invece, i peggiori risultati, si ha che con tolleranze 0.25, 0.05 e 0.01 il metodo meno conveniente è il quattro punti con $\omega = 0.1$. Con $\omega = 0.05$ il metodo a quattro punti risulta il peggiore con tolleranza 0.001. Il metodo non lineare con $\omega = 0.0625$ è il meno conveniente per tolleranze 0.18 e 0.1, con $\omega = 0.05$ risulta il peggiore per tolleranza 0.15.
Per quanto riguarda l'utilizzo di memoria, al contrario della situazione trovata nella compressione di segnali, ci si rende conto che è quasi sempre una buona scelta salvare le matrici in formato sparse. Questo formato risulta essere conveniente con tutti i metodi e con tutte le tolleranze dalla più alta di 0.25 alla tolleranza di 0.08. In particolare i metodi che non utilizzano un parametro definito dall'utente, cioè il due, quattro e sei punti, hanno il numero di bytes impiegati minore anche alla tolleranza 0.05. Quindi, in generale, soltanto se la tolleranza è molto bassa non conviene utilizzare il formato sparse.
Si vuole ora osservare meglio i risultati della ricostruzione delle immagini. Si seguito sono riportati le immagini ricostruite e i valori dei parametri di tolleranza divisi per tolleranza. Nelle seguenti tabelle sono riportati i valori dei parametri di qualità spiegati nel capitolo precedente, quali il PSNR, il Q, con i tre differenti fattori in cui si scompone Q_1, Q_2 e Q_3, e il PEE. I seguenti gruppi di immagini sono divisi per tolleranza e si ha sempre che nella prima riga ci sono gli schemi a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$; nella seconda riga ci sono, invece, gli schemi a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$.

Tolleranza 0.25								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	70.346	70.601	69.819	70.267	70.493	70.200	69.473	69.491
Q_1	0.95	0.95	0.94	0.95	0.95	0.95	0.94	0.94
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
Q	0.945	0.951	0.944	0.946	0.950	0.947	0.932	0.932
PEE	19.836	15.960	22.380	25.795	14.054	11.457	25.759	25.947

Figura 4.36: Tolleranza 0.25 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.18								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	73.058	73.260	71.928	72.791	72.944	72.515	72.438	72.445
Q_1	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	0.971	0.974	0.966	0.971	0.972	0.969	0.967	0.967
PEE	15.611	11.957	15.440	19.317	10.822	7.322	17.144	17.290

Figura 4.37: Tolleranza 0.18 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.15								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	74.589	74.587	73.140	74.215	74.191	73.784	74.174	74.197
Q_1	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	0.980	0.981	0.974	0.979	0.979	0.977	0.978	0.978
PEE	12.610	9.934	10.672	15.629	8.361	4.475	13.575	13.617

Figura 4.38: Tolleranza 0.15 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.1								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	78.099	77.547	76.747	77.394	77.242	77.129	78.187	78.187
Q_1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	0.991	0.990	0.989	0.990	0.990	0.989	0.992	0.992
PEE	7.084	5.085	3.235	8.419	2.634	-0.445	7.520	7.575

Figura 4.39: Tolleranza 0.1 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.08								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	80.292	79.396	78.951	79.178	79.315	79.180	80.206	80.227
Q_1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	0.995	0.994	0.993	0.993	0.994	0.993	0.995	0.995
PEE	4.595	2.643	0.593	5.249	0.750	-1.28	5.366	5.368

Figura 4.40: Tolleranza 0.08 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.05								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	84.955	84.006	83.780	83.586	84.005	83.858	84.635	84.685
Q_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.998
PEE	3.395	0.834	-0.269	1.347	0.207	-1.08	2.865	2.959

Figura 4.41: Tolleranza 0.05 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.01								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	99.407	100.023	102.189	101.030	100.464	101.021	100.336	100.308
Q_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PEE	0.318	0.000	-0.250	-0.162	-0.199	-0.217	-0.043	-0.028

Figura 4.42: Tolleranza 0.01 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.001								
	2pt	4pt	4pt 0.1	4pt 0.05	6pt	8pt	nl 0.0625	nl 0.05
PSNR	128.072	129.451	132.104	130.707	129.957	130.477	129.363	129.270
Q_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PEE	-0.001	-0.001	-0.001	-0.001	-0.003	-0.002	-0.003	-0.002

Figura 4.43: Tolleranza 0.001 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Dalle precedenti immagini e tabelle è possibile osservare i risultati forniti dai parametri di qualità. Osservando soltanto le immagini ci si rende subito conto che diminuendo la tolleranza si hanno dei risultati migliori; osservazione supportata anche dai valori dei parametri. Si nota che il PSNR non è mai nel range di valori medi ($[20, 40]$) ma, anche a

tolleranze alte ha un valore più alto della media. Si nota che diminuendo la tolleranza, il PSNR tende ad aumentare; ciò significa che le immagini ricostruite tendono ad essere sempre più vicine numericamente all' immagine originale della figura 4.35.
Osservando i valori del parametro Q, si ha che sono, anch'essi, abbastanza alti. Si ha, infatti, che i fattori Q_2 e Q_3 rappresentanti le distorsioni della liminosità e del contrasto sono quasi sempre uguali ad 1. Questo significa che non sono presenti questo tipo di distorsioni. Si osserva, però, che sopratutto nelle ricostruzioni con tolleranze alte, i valori del primo fattore Q_1, che rappresenta la correlazione, sono minori di 1; infatti pur non essendoci distorsione le immagini non sono identiche.
Finora si è visto che i parametri PSNR e Q hanno sempre valori accettabili, ma, osservando le immagini, soprattutto nelle figure 4.36, 4.37 e 4.38, si vede che sono sfocate e che non sono di alta qualità. Infatti osservando i valori del parametro PEE, che fornisce una misura proprio della sfocatura di un'immagine, si ha che, per tolleranze alte, sono molto alti, mentre diminuendo la tolleranza si abbassano fino a diventare negativi. In corrispondenza dei PEE negativi, si notano immagini molto verosomiglianti con l'immagine originale.
Importante osservare che il metodo a otto punti fornisce quasi sempre il PEE migliore, anche osservando le immagini ricostruite con questo metodo (le seconde della seconda riga in ogni gruppo) si ha che sono sempre le migliori.
Dalla tolleranza 0.08 si ha che le immagini ricostruite, figura 4.40, iniziano ad essere delle buone ricostruzioni con tutti i metodi utilizzati ed hanno i parametri di qualità con valori alti per il PSNR e per il Q ma bassi abbastanza per il PEE; anche osservando la tabella 4.3 si ha che è conveniente utilizzare il formato sparse per la memorizzazione della matrice delle differenze. Si può dire, allora, che in questo caso la compressione con i metodi di suddivisione risulta una buona scelta.

Si considera, ora, l'immagine a colori della figura 4.44. Come per l'esempio precedente, si applica il metodo di compressione utilizzando i metodi di suddivisione a due, quattro, sei e otto punti; a quattro punti con $\omega = 0.1$ e con $\omega = 0.05$; e il metodo non lineare con $\omega = 0.0625$ e con $\omega = 0.05$. In questo caso occorrono tre diverse matrici per descrivere l'immagine, di conseguenza ci saranno tre diverse matrici delle differenze e tre diversi valori dei parametri di qualità.
Si riporta anche un esempio di immagine a colori per far vedere come si può utilizzare la compressione con gli schemi di suddivisione. Si vuole osservare se questo nuovo metodo risulta essere conveniente e, in caso di risposta affermativa, in quali casi e per quali valori della tolleranza. Inotre si vuole vedere se l'utilizzo del formato sparse per il salvataggio delle matrici delle differenze risulta essere una buona scelta.

Figura 4.44: Immagine originale

Nella seguente tabella sono riportati i risultati ottenuti cambiando i valori alla tolleranza divisi per metodo: sono riportati il numero degli zeri ottenuti e la percentuale rispetto al totale degli elementi per ciascuna metrice, R sta per la matrice dei rossi, G per la matrice dei verdi e B per la matrice dei blu. Inoltre sono riportati i bytes necessari per salvare ciascuna di queste matric delle differenze in formato sparse. Si ha che se si salva l'immagine con il formato usuale si occupano 1,585,175 bytes quindi per ogni matrice il consumo è di 528,392 bytes; questi valori vengono dal fatto che l'immagine ha dimensione 257 x 257 x 3.

Schema a 2 punti									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	56156	**85.02**	55824	**84.52**	53236	**80.60**	*160352*	*165664*	*207072*
0.1	36926	55.91	37495	56.77	35930	54.40	*468032*	*458928*	*83968*
0.05	24962	37.79	25565	38.71	23994	36.33	659456	649808	674944
0.01	7574	**11.47**	8371	**12.67**	7699	**11.66**	937664	924912	935664
0.001	876	**1.33**	1003	**1.52**	880	**1.33**	1044832	1042800	1044768
Schema a 4 punti									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	55334	83.78	55090	83.41	52835	79.99	*173504*	*177408*	*213488*
0.1	37465	**56.72**	37820	**57.26**	36491	**55.25**	*459408*	*453728*	*474992*
0.05	25708	**38.92**	26205	**39.68**	25093	**37.99**	647520	639568	657360
0.01	7324	11.09	7231	10.95	7122	10.78	941664	943152	944896
0.001	154	0.23	154	0.23	150	0.23	1056384	1056384	1056448
Schema a 4 punti con w=0.1									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	51999	78.73	52828	79.98	49068	74.29	*226864*	*213600*	*273760*
0.1	29930	45.31	30599	46.33	28411	43.02	79968	569264	604272
0.05	17259	26.13	17283	26.17	15146	22.93	782704	782320	816512
0.01	3874	5.87	3579	5.42	3215	4.87	996864	1001584	1007408
0.001	433	0.66	371	0.56	351	0.53	1051920	1052912	1053232
Schema a 4 punti con w=0.05									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	52574	79.60	53535	81.05	49492	74.93	*217664*	*202288*	*266976*
0.1	29776	45.08	30464	46.12	28730	43.50	582432	571424	599168
0.05	18458	27.95	18988	28.75	17471	26.45	763520	755040	779312
0.01	4135	6.26	4501	6.81	4111	6.22	992688	986832	993072
0.001	451	0.68	473	0.72	402	0.61	1051632	1051280	1052416
Schema a 6 punti									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	54705	82.82	54914	83.14	52490	79.47	*183568*	*180224*	*219008*
0.1	36445	55.18	36834	55.77	35326	53.48	*475728*	*469504*	*493632*
0.05	25085	37.98	25542	38.67	24212	36.66	657488	650176	671456
0.01	6567	9.94	6644	10.06	6354	9.62	953776	952544	957184
0.001	767	1.16	736	1.11	711	1.08	1046576	1047072	1047472

Schema a 8 punti									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	54813	82.99	55048	83.34	52667	79.74	*181840*	*178080*	*216176*
0.1	36578	55.38	37176	56.29	35113	53.16	*473600*	*464032*	*497040*
0.05	24704	37.40	25252	38.23	23555	35.66	663584	654816	681968
0.01	6363	9.63	6433	9.74	5988	9.07	957040	955920	963040
0.001	713	1.08	745	1.13	703	1.06	1047440	1046928	1047600
Schema non lineare con w=0.0625									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	51962	78.67	53202	80.55	49894	75.54	*227456*	*207616*	*260544*
0.10	29597	44.81	30749	46.55	29084	44.03	585296	566864	593504
0.05	17752	26.88	18105	27.41	16461	24.92	774816	769168	795472
0.01	4860	7.36	4920	7.45	4126	6.25	981088	980128	992832
0.001	510	0.77	525	0.79	445	0.67	1050688	1050448	1051728
Schema non lineare con w=0.05									
Tol	kr	%R	kg	%G	kb	%B	R	G	B
0.25	51986	78.71	53205	80.55	49899	75.55	*227072*	*207568*	*260464*
0.10	29591	44.80	30780	46.60	29074	44.02	585392	566368	593664
0.05	17749	26.87	18082	27.38	16442	24.89	774864	769536	795776
0.01	4859	7.36	4904	7.42	4130	6.25	981104	980384	992768
0.001	514	0.78	527	0.80	456	0.69	1050624	1050416	1051552

Dalla precedente tabella si possono osservare i valori evidenziati in grassetto che indicano i risultati migliori, considerando la percentuale degli zeri maggiore, ottenuti per ogni tolleranza. Si ha che per tolleranza 0.25 risulta essere migliore il metodo a due punti mentre, per tolleranze 0.1 e 0.025, lo schema più conveniente è quello a quattro punti classico. Si nota che diminuendo la tolleranza a 0.01 e a 0.001 si ha che lo schema a due punti risulta, ancora una volta, essere migliore. In questo caso per tutte le matrici delle differenze si ottiene lo stesso risultato.
Si nota, invece, che osservando i peggiori valori, le matrici delle differenze non seguono lo stesso andamento ma forniscono risultati diversi. Considerando la matrice dei rossi, evidenziato in tabella con il colore rosso, si ha che i metodi non lineari risultano essere i peggiori per tolleranze 0.25 e 0.1; diminuendo la tolleranza a 0.05 e a 0.01, lo schema meno conveniente è il quattro punti con $\omega = 0.1$. Infine per la tolleranza minore, 0.001, il metodo peggiore è rappresentato dal quattro punti.
Considerando le matrici delle differenze relative alla matrice dei verdi, evidenziati in tabella in verde, si ha che il metodo peggiore risulta essere il quattro punti con $\omega = 0.1$ per tolleranze 0.25, 0.05 e per 0.01. Per tolleranza 0.1 il meno conveniente è il quattro punti con $\omega = 0.05$ e, per 0.0001 risulta lo schema a quattro punti classico.
Infine, considerando le matrici relative ai blu, in tabella evidenziati in blu, si ha che i peggiori valori si hanno con il metodo a quattro punti con $\omega = 0.1$ per le tolleranze 0.25,

0.1 e 0.01. Per tolleranza 0.05 si ha che il meno conveniente è il metodo non lineare con $\omega = 0.05$ e, per tolleranza 0.001 risulta il metodo a quattro punti classico.
Quindi, per osservare il fenomeno globalmente, si ha che il metodo a quattro punti con $\omega = 0.1$ risulta essere il peggior metodo, mentre lo schema migliore risulta essere il quattro punti classico.
Si vuole, ora, spostare l'attenzione sullo spazio di memoria utilizzato nel caso si utilizzino matrici in formato sparse. I valori convenienti sono evidenziati in tabella in corsivo. Si ricorda che per salvare una singola matrici in formato usuale sono necessari 528,392 bytes. Osservando, quindi, la tabella, si nota che con tolleranza 0.25 conviene utilizzare il formato sparse per ogni metodo; abbassando invece il valore della tolleranza a 0.1 solo i metodi a N punti risultano essere convenienti. Abbassando ulteriormente la tolleranza si ha che con nessun metodo è conveniete salvare le matrici in formato sparse.
Nelle seguenti pagine sono riportate le immagini ricostruite con ogni metodo e divise per tolleranza. In ogni gruppo di immagini si ha, nella prima riga le ricostruzioni con i metodi a due punti, a quattro punti ed a quattro punti con $\omega = 0.1$ e con $\omega = 0.05$; nella seconda riga si hanno gli schemi a sei punti, a otto punti e non lineare con $\omega = 0.0625$ e con $\omega = 0.05$. Nelle tabelle sono riportati i valori dei parametri di qualità per ogni metodo considerato. In particolare sono evidenziati in grassetto i valori del PSNR e del Q riguardanti l'immagine nella sua totalità, calcolati come spiegato nel capitolo precedente nelle formule **?? ??**. Sono presenti, poi, i valori del PSNR, del Q e del PEE per ognuna delle tre matrici; per le matrici dei rossi, dei verdi e dei blu i parametri di qualità sono indicati con le lettere R, G e B al pedice. Per quanto riguarda il metodo Q ci sono anche i valori dei tre fattori in cui si scompone, riportati per ogni matrice.

Figura 4.45: Tolleranza 0.25 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.25				
Metodo	2pt	4pt	4pt w=0.1	4pt w=0.05
PSNR	**68.610**	**68.852**	**67.753**	**67.657**
$PSNR_R$	68.820	69.061	67.945	67.923
$PSNR_G$	68.609	68.845	67.634	67.492
$PSNR_B$	68.410	68.658	67.687	67.568
Q	**0.900**	**0.911**	**0.887**	**0.878**
$Q_{1,2,3}R$	0.88 1.00 0.99	0.89 1.00 1.00	0.86 1.00 1.00	0.86 1.00 0.99
Q_R	0.928	0.936	0.917	0.912
$Q_{1,2,3}G$	0.91 1.00 0.99	0.91 1.00 1.00	0.88 1.00 1.00	0.88 1.00 0.99
Q_G	0.899	0.910	0.882	0.870
$Q_{1,2,3}B$	0.93 1.00 1.00	0.94 1.00 1.00	0.92 1.00 1.00	0.92 1.00 1.00
Q_B	0.873	0.886	0.863	0.852
PEE_R	34.659	29.306	28.450	32.455
PEE_G	33.415	28.716	29.537	34.093
PEE_B	27.116	22.858	19.314	23.827
Metodo	6pt	8pt	nl 0.0625	nl 0.05
PSNR	**68.664**	**68.681**	**67.772**	**67.777**
$PSNR_R$	68.820	68.749	67.893	67.905
$PSNR_G$	68.642	68.723	67.671	67.677
$PSNR_B$	68.535	68.573	67.756	67.753
Q	**0.910**	**0.910**	**0.859**	**0.859**
$Q_{1,2,3}R$	0.89 1.00 1.00	0.89 1.00 1.00	0.87 1.00 0.99	0.87 1.00 0.99
Q_R	0.934	0.932	0.882	0.882
$Q_{1,2,3}G$	0.91 1.00 1.00	0.91 1.00 1.00	0.88 1.00 0.99	0.89 1.00 0.99
Q_G	0.908	0.909	0.912	0.912
$Q_{1,2,3}B$	0.93 1.00 1.00	0.93 1.00 1.00	0.92 1.00 0.99	0.92 1.00 0.99
Q_B	0.887	0.888	0.875	0.875
PEE_R	21.497	14.895	27.074	27.242
PEE_G	22.543	16.290	29.299	29.301
PEE_B	15.734	8.506	20.134	20.267

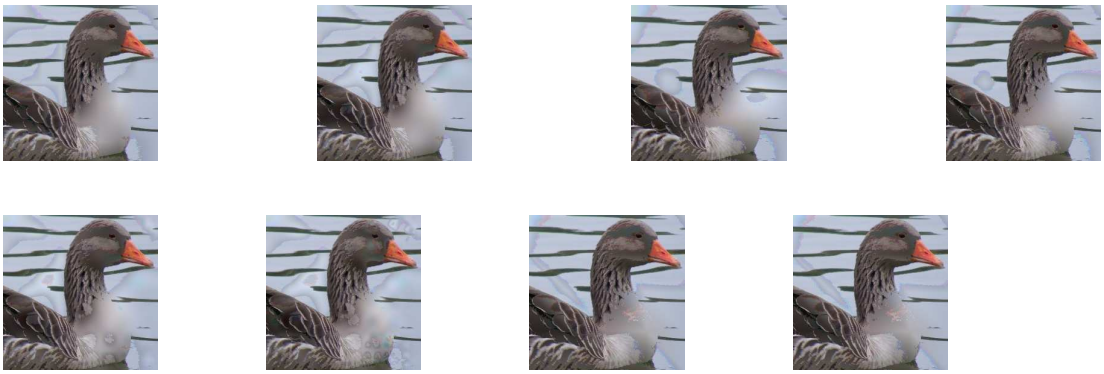


Figura 4.46: Tolleranza 0.1 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.1				
Metodo	2pt	4pt	4pt w=0.1	4pt w=0.05
PSNR	**77.191**	**77.284**	**76.962**	**77.427**
$PSNR_R$	77.224	77.376	76.955	77.534
$PSNR_G$	77.165	77.250	76.874	77.357
$PSNR_B$	77.184	77.226	77.058	77.392
Q	**0.987**	**0.987**	**0.987**	**0.988**
$Q_{1,2,3}R$	0.98 1.00 1.00	0.98 1.00 1.00	0.98 1.00 1.00	0.99 1.00 1.00
Q_R	0.990	0.991	0.990	0.991
$Q_{1,2,3}G$	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00
Q_G	0.987	0.987	0.986	0.987
$Q_{1,2,3}B$	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00
Q_B	0.984	0.985	0.984	0.985
PEE_R	3.620	2.138	-0.314	2.373
PEE_G	3.630	1.987	-0.118	2.521
PEE_B	2.918	1.476	-1.445	1.768
Metodo	6pt	8pt	nl 0.0625	nl 0.05
PSNR	**77.382**	**77.227**	**77.166**	**77.161**
$PSNR_R$	77.482	77.355	77.155	77.165
$PSNR_R$	77.336	77.152	77.051	77.030
$PSNR_R$	77.330	77.178	77.297	77.294
Q	**0.988**	**0.987**	**0.985**	**0.985**
$Q_{1,2,3}R$	0.98 1.00 1.00	0.98 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00
Q_R	0.991	0.991	0.987	0.987
$Q_{1,2,3}G$	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00
Q_G	0.987	0.987	0.990	0.990
$Q_{1,2,3}B$	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00	0.99 1.00 1.00
Q_B	0.985	0.984	0.986	0.986
PEE_R	0.406	-3.113	0.101	0.092
PEE_G	0.008	-3.641	1.027	1.025
PEE_B	-0.303	-4.439	0.306	0.409

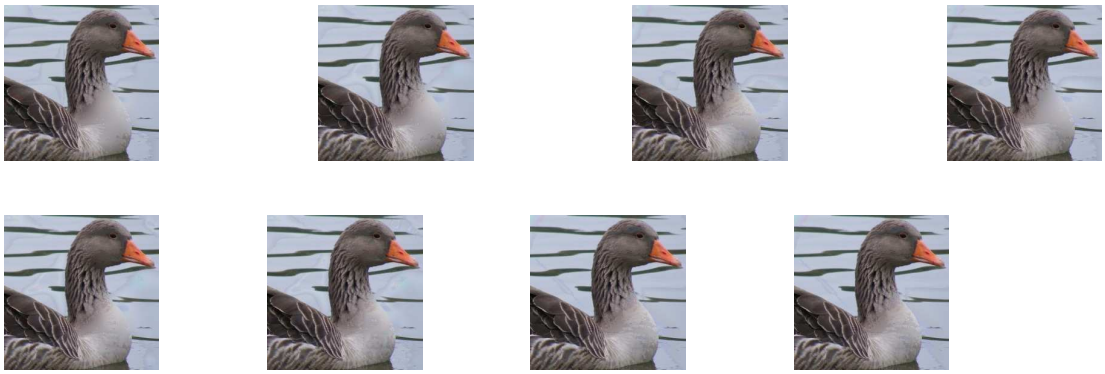


Figura 4.47: Tolleranza 0.05 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.05				
Metodo	2pt	4pt	4pt w=0.1	4pt w=0.05
PSNR	**84.554**	**84.214**	**85.153**	**85.106**
$PSNR_R$	84.677	84.329	85.451	85.333
$PSNR_B$	84.454	84.047	84.903	84.953
$PSNR_G$	84.534	84.272	85.121	85.042
Q	**0.998**	**0.997**	**0.998**	**0.998**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_R	0.998	0.998	0.999	0.998
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	0.998	0.997	0.998	0.998
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	0.997	0.997	0.998	0.997
PEE_R	0.904	-0.170	-1.449	-0.133
PEE_G	0.632	-0.442	-2.196	-0.407
PEE_B	0.503	-0.484	-2.070	-1.013
Metodo	6pt	8pt	nl 0.0625	nl 0.05
PSNR	**84.108**	**84.167**	**85.397**	**85.397**
$PSNR_R$	84.308	84.358	85.464	85.472
$PSNR_G$	83.946	83.973	85.181	85.182
$PSNR_B$	84.078	84.179	85.556	85.544
Q	**0.997**	**0.997**	**0.998**	**0.998**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_R	0.998	0.998	0.998	0.998
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	0.997	0.997	0.999	0.999
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	0.997	0.997	0.998	0.998
PEE_R	-1.174	-2.376	-0.690	-0.683
PEE_G	-1.648	-2.670	-1.304	-1.327
PEE_B	-1.746	-2.755	-1.815	-1.802

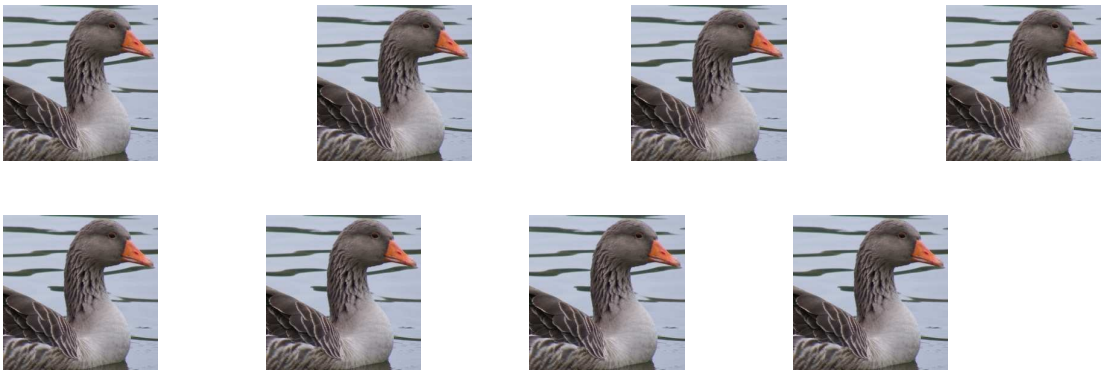


Figura 4.48: Tolleranza 0.01 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.01				
Metodo	2pt	4pt	4pt w=0.1	4pt w=0.05
PSNR	**102.232**	**102.524**	**105.234**	**104.510**
$PSNR_R$	102.045	102.255	105.106	104.170
$PSNR_G$	102.147	102.694	105.320	104.470
$PSNR_B$	102.517	102.636	105.280	104.921
Q	**1.000**	**1.000**	**1.000**	**1.000**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_R	1.000	1.000	1.000	1.000
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	1.000	1.000	1.000	1.000
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	1.000	1.000	1.000	1.000
PEE_R	-0.079	-0.223	-0.118	-0.124
PEE_G	-0.058	-0.194	-0.136	-0.145
PEE_B	-0.035	-0.202	-0.116	-0.147
Metodo	6pt	8pt	nl 0.0625	nl 0.05
PSNR	**102.926**	**103.039**	**104.303**	**104.311**
$PSNR_R$	102.727	102.887	104.266	104.269
$PSNR_G$	102.926	103.131	104.291	104.316
$PSNR_B$	103.135	103.103	104.353	104.348
Q	**1.000**	**1.000**	**1.000**	**1.000**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_R	1.000	1.000	1.000	1.000
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	1.000	1.000	1.000	1.000
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	1.000	1.000	1.000	1.000
PEE_R	-0.228	-0.287	-0.084	-0.080
PEE_G	-0.238	-0.257	-0.063	-0.054
PEE_B	-0.218	-0.236	-0.087	-0.088

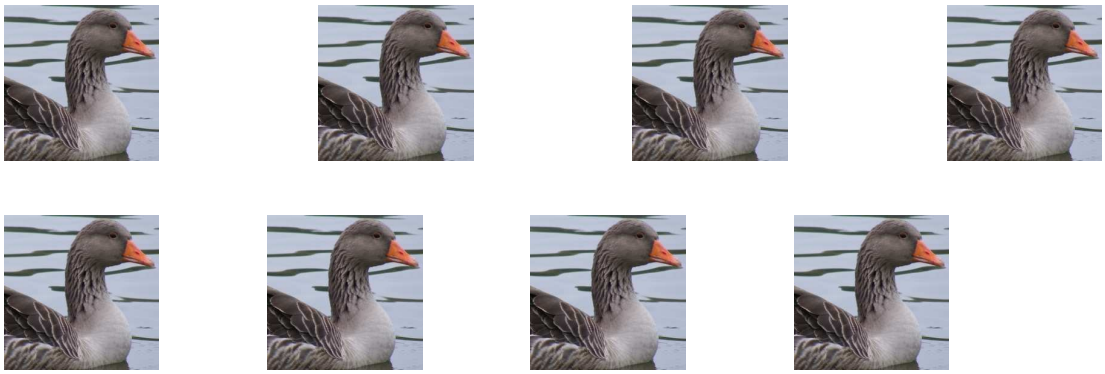


Figura 4.49: Tolleranza 0.001 Sopra: Schema a 2 punti, a 4 punti, a 4 punti con $\omega = 0.1$ e con $\omega = 0.05$ Sotto: Schema a 6 punti, a 8 punti, non lineare con $\omega = 0.0625$ e con $\omega = 0.05$

Tolleranza 0.001				
Metodo	2pt	4pt	4pt w=0.1	4pt w=0.05
PSNR	**116.491**	**116.502**	**116.564**	**116.555**
$PSNR_R$	112.373	112.375	112.401	112.397
$PSNR_G$	120.551	120.634	120.791	120.762
$PSNR_B$	132.228	132.100	135.180	134.727
Q	**1.000**	**1.000**	**1.000**	**1.000**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	1.000	1.000	1.000	1.000
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	1.000	1.000	1.000	1.000
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	1.000	1.000	1.000	1.000
PEE_R	-0.001	-0.000	-0.000	-0.000
PEE_G	-0.002	-0.001	0.000	-0.001
PEE_B	0.000	-0.001	-0.000	-0.001
Metodo	6pt	8pt	nl 0.0625	nl 0.05
PSNR	**116.518**	**116.523**	**116.546**	**116.545**
$PSNR_R$	112.381	112.382	112.393	112.393
$PSNR_R$	120.673	120.679	120.736	120.739
$PSNR_R$	132.872	133.281	134.266	134.073
Q	**1.000**	**1.000**	**1.000**	**1.000**
$Q_{1,2,3}R$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_R	1.000	1.000	1.000	1.000
$Q_{1,2,3}G$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_G	1.000	1.000	1.000	1.000
$Q_{1,2,3}B$	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
Q_B	1.000	1.000	1.000	1.000
PEE_R	-0.001	-0.001	-0.000	-0.001
PEE_G	-0.001	-0.001	-0.001	-0.001
PEE_B	-0.000	-0.001	0.000	-0.000

Come per l'esempio precedente, si può osservare diminuendo la tolleranza si hanno sia immagini migliori che migliori risultati dati dai parametri di qualità.
Osservando il parametro PSNR si ha che, anche in questo caso, ha valori superiori alla media anche per tolleranze alte. Diminuendo la tolleranza i valori crescono e ciò significa che l'immagine ricostruita diventa sempre più vicino numericamente all'immagine originale. Si nota che con tutte le tolleranze fissate, tranne che per 0.05, i metodi a quattro punti classico o con parametro risultano avere i risultati migliori. Si nota poi che per le prime tre tolleranze risulta essere migliore il PSNR della matrice rappresentante i rossi; mentre per tolleranze 0.01 e 0.001, è più alto per le matrici dei blu.
Si nota che il parametro Q è sempre molto alto, diventando proprio 1 nelle ultime due tolleranze. Come per l'esempio precedente, si ha che le immagini ricostruite non presentano distorsioni della luminosità e del contrasto evidenti. Osservando meglio i valori dei fattori Q_1, Q_2 e Q_3 si vede che il valore di Q_1 per tutte le matrici e per tutti i metodi rimane sempre minore rispetto agli altri due valori. Ciò significa che le immagini, pur no avendo distorsioni, non sono linearmente correlate.
Osservando le immagini, però, si nota che, anche se i valori del PSNR e del Q sono accettabili, sono abbastanza sfocate. Questo fenomeno è dovuto ai risultati del parametro PEE. Tali valori, per alte tolleranze, sono molto alti per ogni matrice considerata. Ad diminuire della tolleranza si abbassano fino a diventare negativi in corrispondenza di immagini con un grado di verosomiglianza alto. Anche in questo esempio il metodo che produce il miglior risultato per il PEE è lo schema a otto punti. Infatti si nota che, a parità si tolleranza, le immagini ricostruite con questo metodo (seconda della seconda riga) risultano le migliori.
Avendo ora a disposizione tutti i risultati e tutte le immagini, si possono confrontare i valori ottenuti in queste ultime tabelle con i risultati della tabella **??**. Dalla prima tabella si vede che è conveniente, sia per la percentuale degli zeri presenti nelle matrici delle differenze che per il consumo di bytes, comprimere le immagini con tolleranze alte. Osservando, però, le ricostruzioni e i risultati dei parametri di qualità si ha che, in corrispondenza delle stesse tolleranze, le immagini non sono abbastanza verosimili e che i valori dei parametri non risultano ottimali. In particolare, con tolleranza 0.25 è conveniente utilizzare la compressione con tutti i metodi, salvare le matrici con formato sparse è una buona scelta ma i valori dei parametri di qualità, soprattutto il PEE, non sono accettabili e le ricostruzioni risultano sfocate e troppo distanti dall'immagine originale. Con tolleranza 0.1, le percentuali degli zeri rimangono alte ma utilizzare matrici sparse è conveniente solo per i metodi che non utilizzano un parametro; le immagini ricostruite non sono ancora buone ma migliorano rispetto alla tolleranza più alta, e migliorano anche i valori dei parametri di qualità anche se non sono ancora ottimali. Diminuendo ancora la tolleranza a 0.05 si ha ancora un'accettabile percentuale di zeri ma per nessun metodo è conveniente utilizzare il formato sparse; osservando le immagini, figura 4.45 si vede che sono delle buone ricostruzioni; inoltre i valori dei parametri sono buoni, soprattutto il PEE che, in quasi ogni metodo, risulta essere negstivo. Con questa tolleranza, inoltre,

proprio i valori del PEE raggiungono i livelli migliori, quindi si può affermare che si ha la migliore ricostruzione. Con tolleranze più piccole sia la percentuale degli zeri che i bytes utilizzati dal formato sparse non sono accettabili; le immagini riprodotte sono di altà qualità ma i valori del PEE risultano più alti rispetto alla tolleranza precedente. Ciò significa che nella ricostruzione si aggiungono ulteriori dettagli all'immagine facendo sì che sia leggermente distorta rispetto all'originale.
Si può concludere, allora, che fissando opportune tolleranze ed utilizzando opportuni metodi, la compressione con schemi di suddivisione risulta essere una buona scelta.

Bibliografia

[1] [ZS] D. Zorin and P. Schoröder, *Subdivision for Modeling and Animation* SIGGRAPH 2000 Course Note (2000)

[2] L. Kobbelt, *A variational approach to subdivision* Computer AidedGeometric Design 13: pp. 743-761, 1996

[3] G. Deslauries and S. Dubuc, *Symmetric Iterativ Interpolation Processes* Constr. Approx. 5: pp.49-68, 1989

[4] M.A. Sabin and N.A. Dogson *A Circle-Preserving Variant of the Four-PointSubdivision Scheme*

[5] N. Dyn and D. Levin, *Subdivision schemes in geometric modelling* Acta Numerica (2002), pp. 1-72

[6] N. Dyn, J.A. Gregory and D. Levin, *Analysis of the uniform binary subdivision schemes for curve design* Constr. Approx. 7: pp. 127-147, 1991

[7] N. Dyn and D. Levin, *The Subdivision Experience* An international conference on curves and surfaces on Wavelets, images, and surface fitting, pp. 229-244, July 1994

[8] N. Dyn, *Interpolatory subdivision schemes*, Tutorials on Multiresolution in Geometric Modelling, A. Iske, E.Quak and M.S. Floater (eds), Springer-Verlag, 2002, ISBN 3-540-43639-1, 25-50

[9] N. Dyn, *Linear Subdivison Schemes for the Refinement of Geometric Objects* Proceedings of the International Congress of Mathematicians, Madrid 2006, M. Sanz-Sole, J. Soria, J.L. Varona and J. Verdera (eds.)

[10] N. Dyn *Linear and Nonlinear Subdivision Schemes in Geometric Modeling* Foundations of Computational Mathematics, pp. 68-92 Hong Kong 2008, Cambridge University Press(eds) 2009

[11] R. Schneider and L. Kobbelt *Discrete Fairing of Curve and Surface Based on Linear Curvature Distribution* Curve and Surface Design: Saint-Malo 1999, Laurent, Sablonniere, Schumaker (eds.), pp. 371-380

[12] L. Kobbelt and P. Schröder *A Multiresolution Framework for Variational Subdivision* ACM Transactions on Graphics - TOG , vol. 17, no. 4, pp. 209-237, 1998

[13] J. Warren and H. Weimer *Subdivision Methods for Geometroc Design-A Constructive Approach* Morgan Kaufmann Publishers (2003), chapter 4

[14] T.W. Sederberg, J. Zhang D. Sewell and M. Sabin *Non-Uniform Recursive Subdivision Surface* Proceeding of SIGGRAPH 98, pp 387-394, July 1998.

[15] T.W. Sederberg, J. Zheng and X. Song *Knot Intervals and Multi-Degree Spline* Computer Aided Geometric Design - CAGD , vol. 20, no. 7, pp. 455-468, 2003

[16] L. Romani *From approximating subdivision schemes for exponential splines to high-performance interpolating algorithms* Journal of Computational and Applied Mathematics - J COMPUT APPL MATH , vol. 224, no. 1, pp. 383-396, 2009

[17] C. Beccari, G. Casciola and L. Romani *A non-stationary uniform tension controlled interpolating 4-point scheme reproducing conics*, Computer Aided Geometric Design 24(1), 1-9, 2007

[18] C. Beccari, G. Casciola and L. Romani, *Non-uniform interpolatory curve subdivision with edge parameters built upon compactly supported fundamental splines* BIT Numerical Mathematics, 51(4) (2011), pp. 781-808

[19] D. Levin, *Using Laurent polynomial representation for the analysis of non-uniform binary subdivision schemes* Advances in Computational Mathematics 11, pp. 41-54, 1999

[20] L. Liang, B. Zou, H. Zhao and X. Qiu, *Image interpolation via tangent-control subdivision* Congress on Image and Signal Processing - CISP , 2008

[21] X.D. Liu, S. Osher and T. Chan, *Weighted Essentially Non-Oscillatory Schemes* Journal of Computational Physics, 115 (1994), pp. 200-212.

[22] A. Cohen, N. Dyn and B. Matei, *Quasilinear subdivision schemes with application to ENO interpolation* ACHA 15, pp. 89-116, 2003

[23] M. Dahlen and M. Floater, *Iterative polynomial interpolation and data compression* Numerical Algorithms , vol. 5, no. 3, pp. 165-177, 1993

[24] A. S. Al-Fahoum and A.M. Raza, *Combined Edge Crispiness and Statistical Differencing for Deblocking JPG Compressed Images* IEEE Trans. Image Processing, vol. 10, pp. 1288-1298 Sept. 2001

[25] J. W. Hwang and H. S. Lee, *Adaptive Image Interpolation Based on Local Gradient Features* IEEE Signal Processing Letters, vol. 11, no 3, pp. 359-362 March 2004

[26] Z.Wangand A.C.Bovik,*A Universal Image Quality Index*, IEEE Signal processing Letters,Vol.9,pp. 81-84,March 2002.

[27] A. Horé and D. Ziou*Image quality metrics: PSNR vs. SSIM*, International Conference on Pattern Recognition 2010.