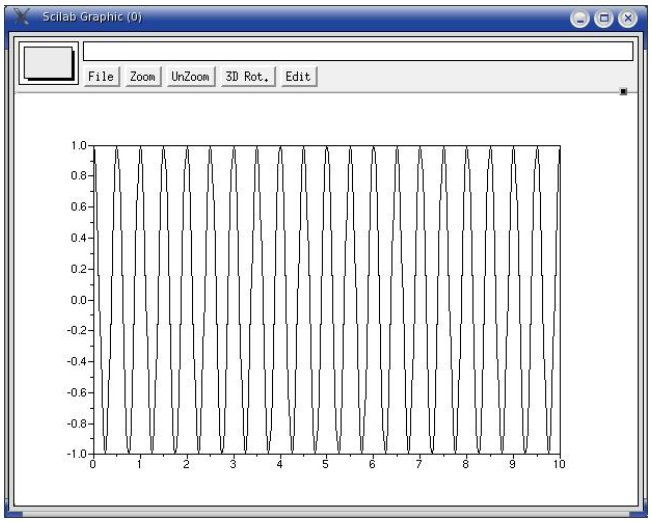
Représentation graphique : échantillonnage

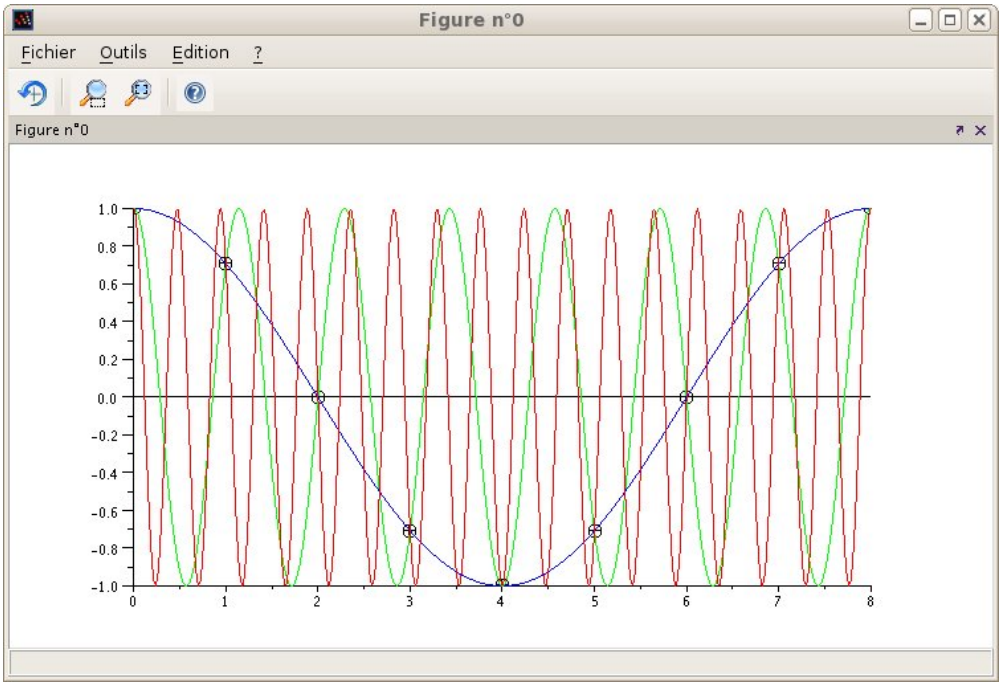
La fonction à représenter, $f(t) = cos(4\pi\, t)$, a de fortes variations : sa période est $T = 1/2$.
Comme le pas `x(i+1)-x(i)`$= 0.5$, la fonction semble constante !
Il faut donc échantillonner suffisamment l'intervalle $[0, 1]$ pour obtenir un graphe acceptable !

```
--> x = linspace(0,10,401); y = cos(4*%pi*x);
--> scf(0); plot2d(x,y)
```



Échantillonnage-Interpolation

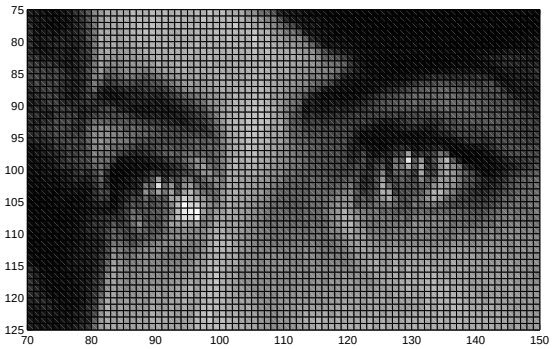
On considère 9 points `x = cos(%pi*[0:8]./4)` :



Les valeurs discrètes $x(n)$ peuvent provenir de $\cos(\frac{\pi}{4}\, t)$, de $\cos(\frac{7\pi}{4}\, t)$ de $\cos(\frac{17\pi}{4}\, t)$ ou...

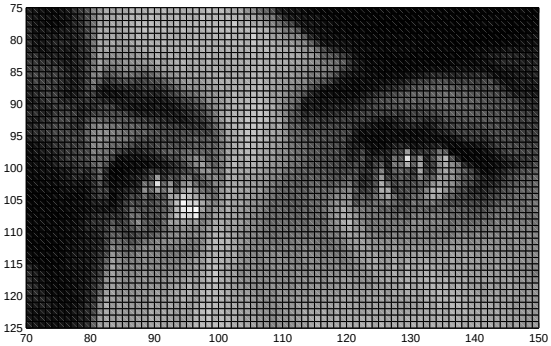
Images numériques

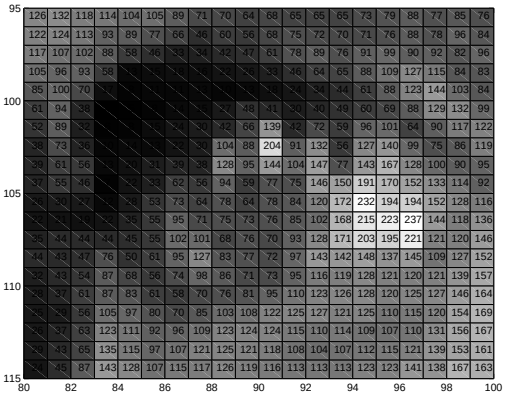
La commande `Matplot` permet d'afficher des matrices en tant qu'images numériques : un entier entre 0 et 255, codé sur un octet, est associé à une luminance.
Ainsi 0 est "noir" et 255 est "blanc" :
le pixel (l, c) est affiché au niveau de gris $Im(l, c) \in \{0, \dots 255\}$.
Attention : l'origine $(l, c) = (0, 0)$ se trouve en haut à gauche !



Images numériques

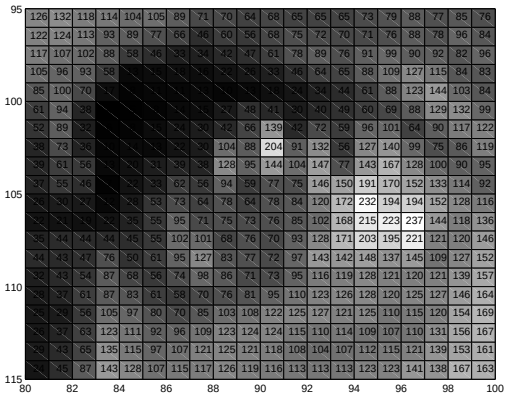
La commande `Matplot` permet d'afficher des matrices en tant qu'images numériques : un entier entre 0 et 255, codé sur un octet, est associé à une luminance.
Ainsi 0 est "noir" et 255 est "blanc" :
le pixel (l, c) est affiché au niveau de gris $Im(l, c) \in \{0, \dots 255\}$.

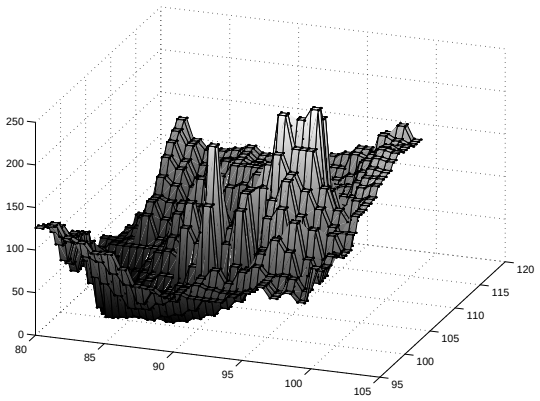




Images numériques

La commande `Matplot` permet d'afficher des matrices en tant qu'images numériques : un entier entre 0 et 255, codé sur un octet, est associé à une luminance.
Ainsi 0 est "noir" et 255 est "blanc" :
le pixel (l, c) est affiché au niveau de gris $Im(l, c) \in \{0, \ldots 255\}$.
Si on pose $z = Im(l, c)$, on représente une *surface* : $(l, c, Im(l, c))$.





Images numériques : inversion vidéo

Une opération très simple sur les niveaux de gris est l'inversion vidéo de l'image :

$$Im_{inv\_video}(l, c) = 255 - Im(l, c) \ .$$

Images numériques : changement de contraste

On utilise une transformation croissante des niveaux de gris, par exemple :

$$Im_{c(\gamma)}(l, c) = 255 * (Im(l, c)/255)^{\gamma} ,$$

où $\gamma \in \mathbb{R}_+^*$.

$\gamma = 1/2$

$\gamma = 2$

Images numériques : symétries/miroirs

On peut aussi agir sur le domaine de la fonction $Im(l, c)$, par exemple :

$Im_{gd}(l, c) = Im(l, NbCol+1-c)$ et $Im_{bh}(l, c) = Im(NbLigne+1-l, c)$

Images numériques : Rotation

Une rotation de 90° est facile avec Scilab : Im_{90°
mais une rotation de 60° est plus compliquée à mettre en oeuvre !

Images numériques : compression

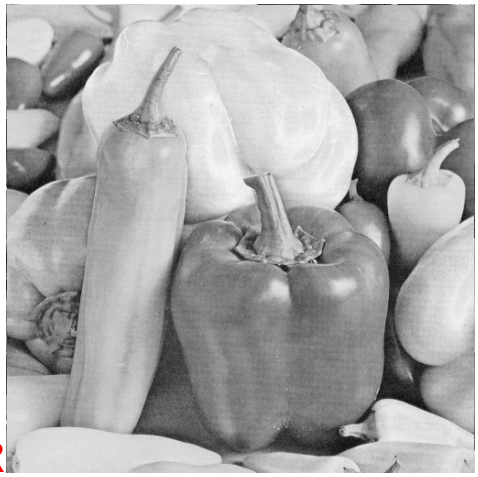
La taille de l'image test est $2^8 \times 2^8 = 256 \times 256$,
on peut donc avoir une partition en blocs de taille $2^p \times 2^p$ pixels.
Pour $p = 3$, on fait la moyenne sur les blocs 8×8,
il ne reste que $2^5 \times 2^5 = 32 \times 32$ pixels.
On a un taux de compression de *taille finale/taille initiale*$= 1/64$!

Images numériques : décompression

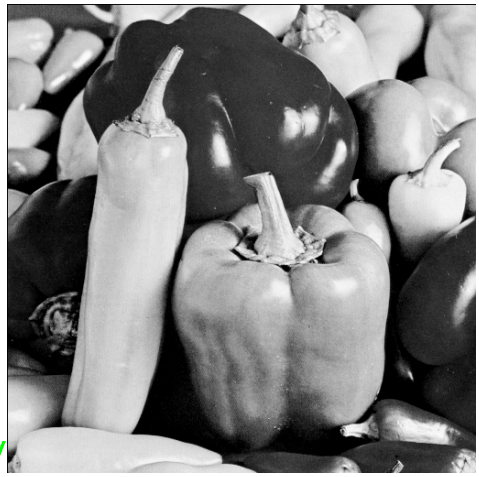
Pour revenir à l'image initiale, on fait l'opération inverse :
un pixel donne un bloc constant de taille $2^p \times 2^p$.
Mais la reconstruction n'est pas très jolie ! Des standards de compression comme `JPEG` utilisent des idées plus sophistiquées.

$p = 3$

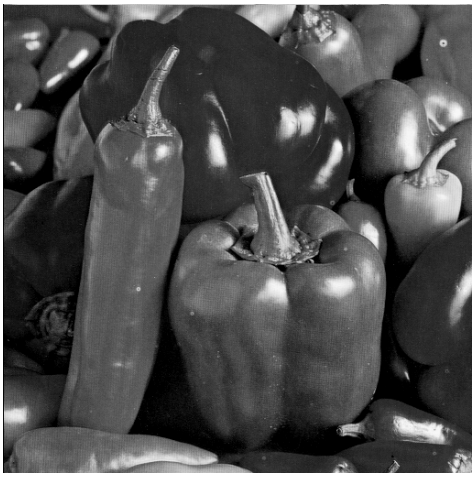
Pour représenter une image en c o u l e u r s
on code un pixel sur trois canaux : (Rouge,Vert,Bleu)



R



V



B