Numerisk Analys, MMG410. Lecture 3.

Flyttalsaritmetik

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

var

$$0 \leq d_k \leq \beta - 1, L \leq e \leq U,$$

Här:

- β bas (s vi kommer att ha $\beta = 2$)
- e exponent (heltal)
- t precision
- $[L, U]$ exponentomfång
- $d_k, k = 0, ..., t - 1$ mantissa (heltal)

Vi antar att $= 2$ från och med nu:

bas	t	L	U	
2	24	-126	127	32 bitar
2	53	-1022	1023	64 bitar

Example

$-9.28 := -[1.16] \cdot 2^3 = -[1 + 0.16] \cdot 2^3$

$$0.16 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \frac{1}{32}x_5 + \ldots =$$

$$= \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 0 + \ldots$$

Exponenten 3 lagras som: $3 + 1023 = 1026 = 1024 + 2 = 1 \cdot 2^{10} + 1 \cdot 2^1 + 0 \cdot 2^0$

1	10000000010	00101 ....
tecken	exponent 11 bitar	mantissa 52 bitar

$\frac{1}{2} = 0.5 > 0.16$, darför $\frac{1}{2} \cdot 0$, $\frac{1}{4} = 0.25 : 0.25 > 0.16$, darför $\frac{1}{4} \cdot 0$, $\frac{1}{8} = 0.125 < 0.16$ och darför $\frac{1}{8} \cdot 1$, $\frac{1}{16} = 0.0625 : 0.16 - 0.125 = 0.035, 0.0625 > 0.035$, och darför $\frac{1}{16} \cdot 0$, $\frac{1}{32} = 0.0312 : 0.0312 < 0.035$, och darför $\frac{1}{32} \cdot 1$, och så vidare ...

Example

$-9.28 := -[1.16] \cdot 2^3 = -[1 + 0.16] \cdot 2^3$

Kollar i Matlab:

```
q = quantizer('double');
y = num2bin(q,-9.28)
y =
1100000000100010100011110101110000101000111101011100001010001
```

Obs: uppe finns inte plats for y på sliden, vi ska ha:

```
y =
11000000001000101000111101011100001010001111010111000010100011111
```

Example

$6.28 = +[1.57] \cdot 2^2 = +[1 + 0.57] \cdot 2^2$

$$0.57 = \frac{1}{2} + 0.07 =$$

$$= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} + 0 \cdot \frac{1}{32} + \ldots + \frac{1}{256} + \ldots$$

Exponenten 2 lagras som: $2 + 1023 = 1025 = 1024 + 1 = 1 \cdot 2^{10} + 1 \cdot 2^0$

0	10000000001	10010 ....10...
tecken	exponent 11 bitar	mantissa 52 bitar

I Matlab:

y = num2bin(q,6.28)

y =

0100000000011001000111101011100001010001111010111000010100011

Syns inte sista siffror: y =

0100000000011001000111101011100001010001111010111000010100011111

Example

$1 := [1] \cdot 2^0 = [1 + 0.0] \cdot 2^0$

Mantissa är 0 här.

Exponenten 0 lagras som: $0 + 1023 = 1024 - 1 = 1 \cdot 2^{10} - 1 \cdot 2^0 =$

$\underbrace{10000000000}_{11\ bitar} - \underbrace{00000000001}_{11\ bitar} = \underbrace{01111111111}_{11\ bitar}$.

0	01111111111	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Example

$0.5 := [1] \cdot 2^{-1} = [1 + 0.0] \cdot 2^{-1}$

Mantissa är 0 här.

Exponenten -1 lagras som: $-1 + 1023 = 1024 - 2 =$

$1 \cdot 2^{10} - 1 \cdot 2^{1} = \underbrace{10000000000}_{11\ \textit{bitar}} - \underbrace{00000000010}_{11\ \textit{bitar}} = \underbrace{01111111110}_{11\ \textit{bitar}}.$

0	01111111110	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Example

$0.1 := [1.6] \cdot 2^{-4} = [1 + 0.6] \cdot 2^{-4}$

Exponenten -4 lagras som:

$-4 + 1023 = 1019 = 1024 - 5 = 1 \cdot 2^{10} - (1 \cdot 2^2 + 1 \cdot 2^0) =$

$\underbrace{10000000000}_{11\ bitar} - \underbrace{00000000101}_{11\ bitar} = \underbrace{01111111011}_{11\ bitar}.$

0	01111111011	100...
tecken	exponent 11 bitar	mantissa 52 bitar

Kollar i Matlab:

```
q = quantizer('double');
y = num2bin(q,0.1)
y =
0011111110111001100110011001100110011001100110011001100110011
```

Obs: uppe finns inte plats for y på sliden, vi ska ha:

```
y =
0011111110111001100110011001100110011001100110011001100110011010
```

Flyttal-räkning

Om x är ett godtyckligt reellt tal betecknar vi det avrundade flyttalet med $fl(x)$ (floating). Normalt (kan ändras) är fl(x) det flyttal som ligger närmast x.

Exempel:

Låt oss anta att vi räknar decimalt med fyra siffror.

$$\begin{aligned} fl(\pi) &= fl(3.141592653589...) = 3.142, \\ fl(31415926.53589...) &= 3.142 \cdot 10^7. \end{aligned} \tag{1}$$

Hur stort kan det absoluta felet bli vid avrundning till närmaste flyttal? Maximalt en halv enhet i fjärde siffran. Så om vårt tal är

$$\pm s_1 . s_2 s_3 s_4 ... 10^e$$

(där $s_1, s_2, ...$ betecknar decimala siffror) är absolutbeloppet av absoluta felet maximalt $0.0005 \cdot 10^e$. Relativa felet är maximalt (för ett normaliserat tal)

$$\left| \frac{fl(x) - x}{x} \right| \leq \frac{0.0005 \cdot 10^e}{1.0000.... \cdot 10^e} = 0.0005$$

Denna begränsning kallas relativa maskinnoggrannheten

$\max \varepsilon_{mach} := 0.0005$

Maskinnoggrannheten ε_{mach}

ε_{mach} beror på method, som vi använder i avrundning. Definitioner för ε_{mach} för precision t:

- I rounding by choping:

$$\varepsilon_{mach} = \beta^{1-p} = \beta^{-t}$$

- I rounding to nearest:

$$\varepsilon_{mach} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}\beta^{-t}$$

I dubbel precision (när $t = 53$, 64 bits dator) gäller att $\varepsilon_{mach} = 2^{-t} \approx 1.11 \cdot 10^{-16}$ och i enkel precision (när $t = 24$, 32 bits dator) $\varepsilon_{mach} = 2^{-t} \approx 6 \cdot 10^{-8}$.

Example

	chop rounding to 2 digits	to nearest to 2 digits
1.849	1.8	1.8
1.850	1.8	1.9
1.851	1.8	1.9
1.899	1.8	1.9

Now the following values of machine epsilon apply to standard floating point formats:

Table 1. *Values of machine epsilon in standard floating point formats. Notation $*$ means that one bit is implicit in precision p. Machine epsilon ε_{mach} is computed accordingly to Demmel, Applied Numerical Linear Algebra, SIAM.*

EEE 754 - 2008	description	Base, b	Precision, p	Machine eps. $\varepsilon_{mach} = 0.5 \cdot \beta^{-(p-1)}$
binary16	half precision	2	11^*	$2^{-11} = 4.88e-04$
binary32	single precision	2	24^*	$2^{-24} = 5.96e-08$
binary64	double precision	2	53^*	$2^{-53} = 1.11e-16$
binary80	extended precision	2	64	$2^{-64} = 5.42e-20$
binary128	quad. precision	2	113^*	$2^{-113} = 9.63e-35$
decimal32	single prec. decimal	10	7	5×10^{-7}
decimal64	double prec. decimal	10	16	5×10^{-16}
decimal128	quad. prec. decimal	10	34	5×10^{-34}

Flyttal-räkning

Normaliserat tal:

$$\pm s_1.s_2 s_3 s_4 ... 10^e$$

,
där $s_1, s_2, .., s_1 \neq 0$. betecknar decimala siffror.
Om x är ett godtyckligt reellt tal betecknar vi det avrundade flyttalet med $fl(x)$ (floating). Normalt (kan ändras) är fl(x) det flyttal som ligger närmast x.
Denormaliserat tal i enkel precision (f=fraktion):

$$\pm 0.f \cdot 2^{-126}.$$

Denormaliserat tal i dubbel precision:

$$\pm 0.f \cdot 2^{-1022},$$

Alla icke-zero reella tal kan normaliseras.

Example

$x = 918.082$ i normalizerat form: $9.18082 \cdot 10^2$.
-0.00574012 i normalizerat form: $-5.74012 \cdot 10^{-3}$.

Flyttal-räkning: IEEE Standard 754

Floating Point Range

	Denormaliserat	Normaliserat
Enkel prec.	$\pm 2^{-149}$ till $(1 - 2^{-23}) \cdot 2^{-126}$	$\pm 2^{-126}$ till $(2 - 2^{-23}) \cdot 2^{127}$
Dubbel prec.	$\pm 2^{-1074}$ till $(1 - 2^{-52}) \cdot 2^{-1022}$	$\pm 2^{-1022}$ till $(2 - 2^{-52}) \cdot 2^{1023}$

	Decimal
Enkel prec.	$\pm \approx 10^{-44.85}$ till $\approx 10^{38.53}$
Dubbel prec.	$\pm \approx 10^{-323.3}$ till $\approx 10^{308.3}$

Maskinnoggrannheten ε_{mach}

Vi kan skriva

$$\left|\frac{fl(x) - x}{x}\right| \leq \varepsilon_{mach}$$

på ett annat sätt. Det gäller med $|\varepsilon| \leq \varepsilon_{mach}$ att

$$fl(x) = (1 + \varepsilon)x = x + \varepsilon x$$

Varför gäller detta? Antag att $x \neq 0$

$$fl(x) = (1 + \varepsilon)x;$$
$$fl(x) - x = \varepsilon x;$$
$$\underbrace{\left|\frac{fl(x) - x}{x}\right|}_{\leq \varepsilon_{mach}} = |\varepsilon|$$

Detta gäller även om $x = 0$, då $fl(x) = 0$.

Flyttal-räkning

Låt $\otimes$ beteckna någon av operationer $+, -, *, /$. Låt x och y vara två flyttal. Då gäller att med $|\varepsilon| \leq \varepsilon_{mach}$:

$$fl(x \otimes y) = (1 + \varepsilon)(x \otimes y) = (x \otimes y) + \varepsilon(x \otimes y)$$

Beloppet av absoluta felet är:

$$|fl(x \otimes y) - (x \otimes y)| = |\varepsilon(x \otimes y)| \leq \varepsilon_{mach}|x \otimes y|$$

och relativa felet är (om $x \otimes y \neq 0$)

$$\frac{|fl(x \otimes y) - (x \otimes y)|}{|x \otimes y|} = |\varepsilon| \leq \varepsilon_{mach}.$$

Flyttal-räkning: problem

Några vanliga problem med fluttalsräkning:
Antag att vi räknar i dubbel precision:

1e16 + 1 - 1e16
ans = 0

men om vi skriver

1e16 - 1e16 + 1
ans = 1

Flyttal-räkning: problem

1 + 1e-16 - 1
ans = 0

men

1 -1 + 1e-16
ans = 1.0000e-16

Flyttal-räkning: problem

1e16 + 1.00000000001 - 1e16
ans = 2

men

1e16 - 1e16 + 1.00000000001
ans = 1

Flyttal-räkning: kancellation

Man bör undvika att addera eller subtrahera tal av mycket olika storleksordning.
$a + (b + c)$ behöver inte vara lika med $(a + b) + c$.
En kompilator får inte optimera för mycket.
Kancellation - subtraktion av två nästan lika stora tal.
Antag att vi subtraherar två olika tal:

1.03678947f
1.03678935g
==========
0.00000012t

Här, f och g betecknar fel, och fi får nytt fel t. I de två första talen kommer felet i tionde siffran: $1.03678947 \underbrace{f}_{10\ siffra}$, och i skillnaden finns felet redan i tredje sifrran: $0.00000012t = 1.2 \underbrace{t}_{3\ siffra} e - 7$. Vi har förlorat information.

Flyttal-räkning: exempel

Antag att a, b, c är redan avrundade flyttal och att vi vill beräkna $a + bc$. Vi får införa en ε-term för varje räkneoperation med $|\varepsilon_k| \leq \varepsilon_{mach}, k = 1, 2$:

$$fl(a + bc) = fl(a + fl(bc)) = (a + bc(1 + \varepsilon_1))(1 + \varepsilon_2).$$

Multiplicera ihop faktorerna och tar beloppet av absoluta felet:

$$fl(a + bc) = a + bc + bc\varepsilon_1 + (a + bc)\varepsilon_2 + bc\varepsilon_1\varepsilon_2,$$
$$|fl(a + bc) - (a + bc)| = |bc\varepsilon_1 + (a + bc)\varepsilon_2 + bc\varepsilon_1\varepsilon_2|.$$

Övre begränsning av det absoluta felet är:

$$|fl(a + bc) - (a + bc)| \leq |bc|\varepsilon_{mach} + |a + bc|\varepsilon_{mach} + |bc|\varepsilon_{mach}^2|$$
$$= ((1 + \varepsilon_{mach})|bc| + |a + bc|)\varepsilon_{mach}$$

Flyttal-räkning: exempel

För relativa felet har vi (om $a + bc \neq 0$):

$$\frac{|fl(a+bc) - (a+bc)|}{|a+bc|} \leq \frac{(1+\varepsilon_{mach})|bc| + |a+bc|}{|a+bc|}\varepsilon_{mach}$$
$$= \left[\frac{(1+\varepsilon_{mach})|bc|}{|a+bc|} + 1\right]\varepsilon_{mach}.$$

Relativa felet är litet om $\frac{|bc|}{|a+bc|}$ inte är för stort. Om, till exempel, $|bc| \approx 1, |a+bc| \approx 0$, vi få ett stort relativt fel.
Om man inte kräver en strikt gräns utan endast en uppskattning kan man tillåta sig att släga t.ex. $\varepsilon_1\varepsilon_2$- termer (produkter av termer) ty $\varepsilon_{mach}^2 << \varepsilon_{mach}$.

Flyttal-räkning: exempel

För att se att analysen stämmer rätt bra kommer här ett numeriskt exempel i fyrsiffrig decimal aritmetik:

$$a = 10.70,\ b = -4.567,\ c = 2.344,\ a + bc = -0.005048\ (\textit{exakt}).$$

$$\begin{aligned} fl(a + bc) &= fl(a + fl(bc)), \\ fl(bc) &= -10.71\ (\text{eftersom } bc = -10.705048), \\ fl(a + fl(bc)) &= fl(10.70 + (-10.71)) = -0.010. \end{aligned}$$

Beloppet av absoluta felet är:

$$\begin{aligned} a + bc &= -0.005048\ \text{exakt}, \\ |fl(a + bc) - (a + bc)| &= |-0.010 - (-0.005048)| = 0.004952. \end{aligned}$$

Notera att detta fel är ungefär lika stort som det exakta värdet. Det relativa felet är:

$$\frac{|fl(a + bc) - (a + bc)|}{|a + bc|} = \frac{|-0.010 - (-0.005048)|}{|-0.005048|} \approx 0.98.$$

Flyttal-räkning: exempel

Det relativa felet är:

$$\frac{|fl(a+bc)-(a+bc)|}{|a+bc|} = \frac{|-0.010-(-0.005048)|}{|-0.005048|} \approx 0.98$$

$$\leq \underbrace{\left[\frac{(1+\varepsilon_{mach})|bc|}{|a+bc|}+1\right]\varepsilon_{mach}}_{uppskattning\ \ för\ \ felet}$$

$$= \left[\frac{(1+0.0005)|\cdot|-10.705048|}{|-0.005048|}+1\right] 0.0005 \approx 1.061$$

Här, relativa maskinnoggrannheten $\max \varepsilon_{mach} := 0.0005$.
Resultat stämmer väl med uppskattning, vi fick: $0.98 \leq 1.061$.

Matrisfaktoriseringar: LU-faktorisering

Vanligt i tillämpningar och teoretiskt arbete att skriva matriser som produkter av andra matriser (kallas matrisfaktoriseringar eller uppdelningar). Några exempel illustrerade med små "kryssmatriser":

$$\underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} x & 0 & 0 \\ x & x & 0 \\ x & x & x \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix}}_{U}$$

- L för Lower triangular, undertriangulär och
- U för Upper triangular, övertriangulär.
- Kallas LU-faktorisering. Används för att lösa Ax = b-problem.
- Matlab-kommando **lu**.

Matrisfaktoriseringar: QR-faktorisering

För att approximativt lösa överbestämda ekvationssystem (minstakvadratproblem) använder vi QR-faktorisering:

$$\underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix}}_{R}$$

- $\dim A = m \times n$, $\dim Q = m \times n$, $\dim R = n \times n$.
- Q för ortogonal matris, dvs. $Q^T Q = I_n$.
- R för Upper triangular, övertriangulär.
- Kallas QR-faktorisering. Används för att lösa minstakvadratproblem, hitta egenvärden i symmetrisk matris.
- Matlab-kommando **qr**.

Matrisfaktoriseringar: diagonalisering

Om A är en s.k. diagonaliserbar matris kan vi använda Matlabs **eig**-kommando för att beräkna:

$$\underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{X} \cdot \underbrace{\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}}_{\Lambda} \cdot \underbrace{\begin{bmatrix} x & 0 & 0 \\ x & x & 0 \\ x & x & x \end{bmatrix}}_{X^{-1}}$$

$\lambda_1, \lambda_2, \lambda_3$ är A:s egenvärden och de tre kolonnerna i X är motsvarande egenvektorer. Om A är en reell och symmetrisk matris så kan egenvektorerna väljas ortonormerade varför X är ortogonal och $X^{-1} = X^T$.

Singulära värdena

Låt A vara en $m \times n$ matris och låt $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ vara egenvärdena till A^*A ordnade i storleksordning. Talen $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ definierade genom $\sigma_j = \sqrt{\lambda_j}$ kallas de singulära värdena till A.
Conjugate transpose matrix:

$$A^*_{ij} = \overline{A}_{ji},$$

$\overline{A}_{ji}$ - scalar complex conjugate elements.

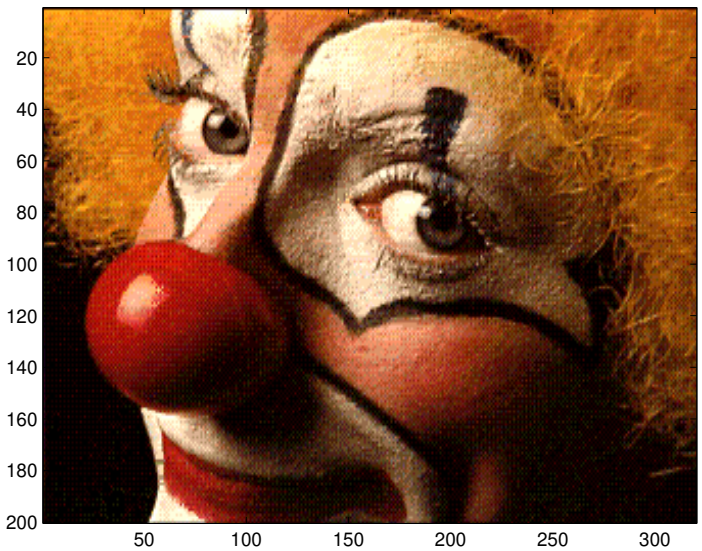
Singuläravärdesfaktoriseringen (SVD)

Singuläravärdesfaktoriseringen (i Matlab commando **svd**) är en slags generalisering av egenvärdesuppdelningen av $A = U\Sigma V^T$ till ickekvadratiska matriser.

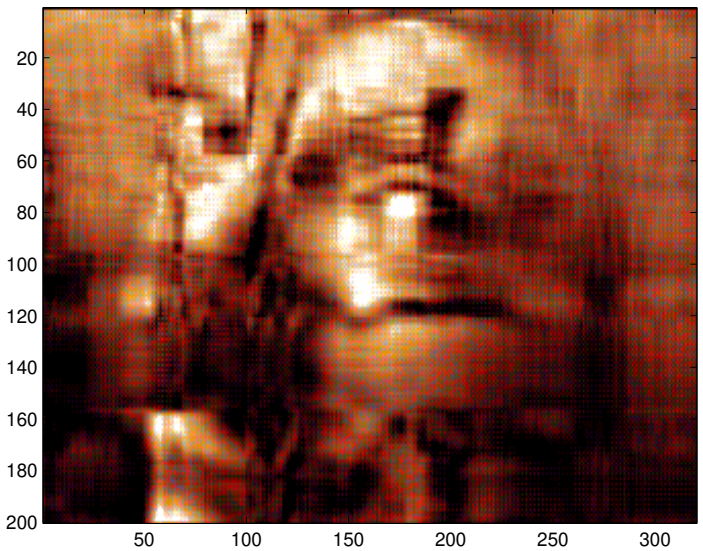
$$\underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}}_{U} \cdot \underbrace{\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\Sigma} \cdot \underbrace{\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}}_{V^T}$$

där $U^T U = I, V^T V = I$ är ortogonala matriser och $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Faktoriseringen existerar även för liggande matriser. Används för att lösa minstakvadratproblem, hitta egenvärden i symmetrisk matris, komprimera bilder.

Application SVD: Image compression using SVD



a) Original image



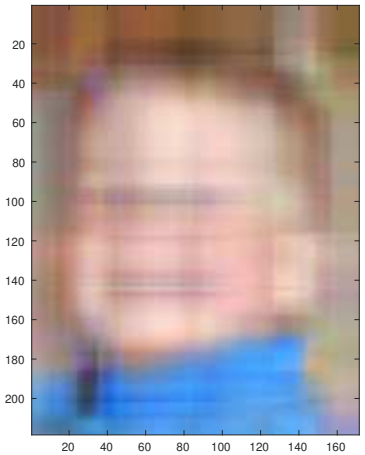
b) Rank k=20 approximation

Image compression using SVD in Matlab

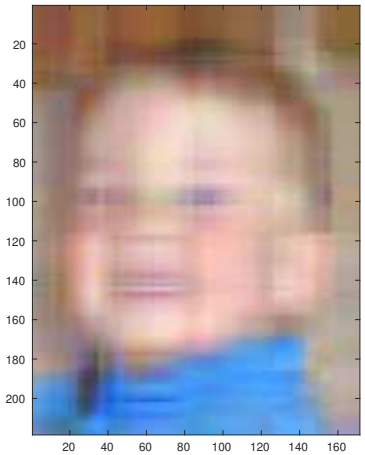
See path for other pictures:
/matlab-2012b/toolbox/matlab/demos
load clown.mat;
Size(X) $= m \times n = 320 \times 200$ pixels.
[U,S,V] = svd(X);
colormap(map);
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)$= m \times k$, size(V)$= n \times k$.

Image compression using SVD in Matlab

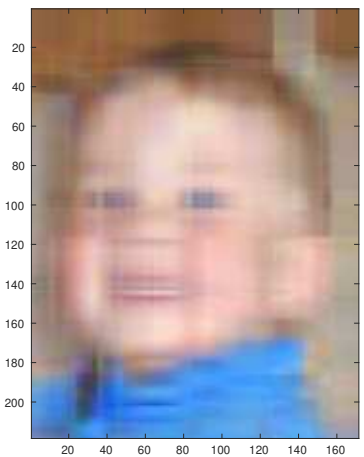
a) Original image



b) Rank k=4 approximation



b) Rank k=5 approximation



c) Rank k=6 approximation

d) Rank k=10 approximation

d) Rank k=15 approximation

Image compression using SVD for arbitrary picture

To get image on the previous slide, I took picture in jpg-format and loaded it in matlab. You can also try to use following matlab code for your own pictures:

```
A = imread('Child.jpg');
Real size of A: size(A) ans= 218 171 3
figure(1); image(DDA);
DDA=im2double(A);
[U1,S1,V1] = svd(DDA(:,:,1)); [U2,S2,V2] = svd(DDA(:,:,2));
[U3,S3,V3] = svd(DDA(:,:,3));
k=15;
svd1 = U1(:,1:k)*S1(1:k,1:k)*V1(:,1:k)';
svd2 = U2(:,1:k)*S2(1:k,1:k)*V2(:,1:k)';
svd3 = U3(:,1:k)*S3(1:k,1:k)*V3(:,1:k)';
DDAnew = zeros(size(DDA));
DDAnew(:,:,1) = svd1; DDAnew(:,:,2) = svd2; DDAnew(:,:,3) = svd3;
figure(2); image(DDAnew);
```