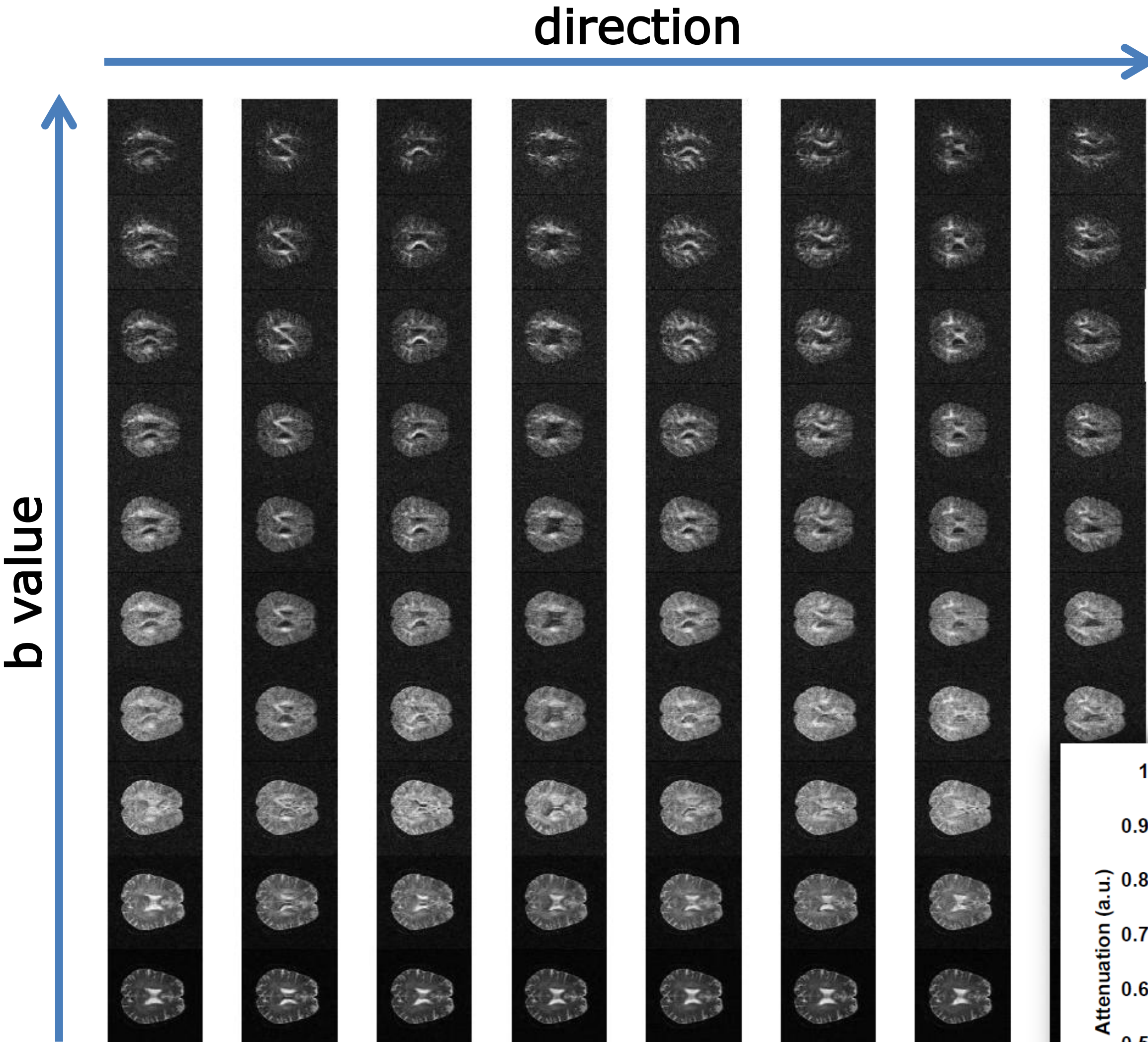
Anno 2012
Protocollo: RBFR12JF2Z

Unità A2: Imaging NMR, Roma

Accelerazione della ricostruzione di immagini NMR ottenute con il metodo del contrasto in diffusione

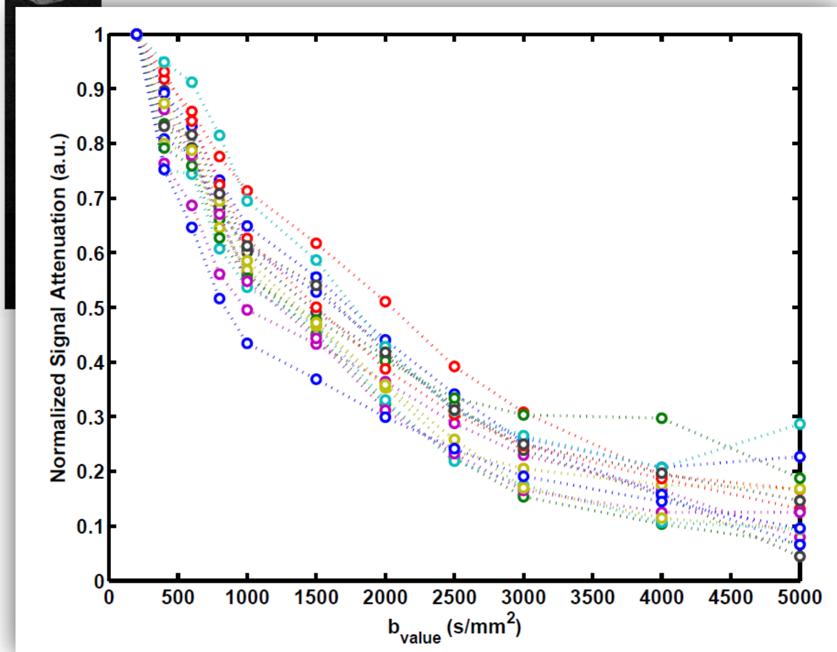
Dr.ssa **Silvia Capuani**[*$], Dr. **Marco Palombo**[*$]

*Physics Department, "Sapienza" University of Rome,
$ CNR-IPCF UOS Roma, Rome



Typical NMR imaging post-processing

128x128 pixel image
x
number of slice (~10-40)
x
number of diffusion gradient directions (≥15)



Cumulant expansion approach: kurtosis tensor measurement

$$S(b) \propto e^{-D_{app} b + \frac{1}{6} K_{app} [D_{app} b]^2 + \ldots}$$

$$S(b) = S(0) \exp\left[-b \sum_{i,j} n_i n_j D_{ij} + \frac{1}{6} b^2 \left(\frac{1}{3} \sum_i D_{ii} \right)^2 \sum_{i,j,k,l} n_i n_j n_k n_l W_{ijkl} + O(b^3) \right]$$

N ≥ 15 independent measures to get W_{ijkl}

$$K(\mathbf{n}) = \frac{\overline{D}^2}{[D(\mathbf{n})]^2} \sum_{i,j,k,l=1}^{3} n_i n_j n_k n_l W_{ijkl}.$$

$$MK = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{n}_i)$$

$$\overline{K} = F_1(\lambda_1, \lambda_2, \lambda_3)\tilde{W}_{1111} + F_1(\lambda_2, \lambda_1, \lambda_3)\tilde{W}_{2222} + F_1(\lambda_3, \lambda_2, \lambda_1)\tilde{W}_{3333} + F_2(\lambda_1, \lambda_2, \lambda_3)\tilde{W}_{2233} + F_2(\lambda_2, \lambda_1, \lambda_3)\tilde{W}_{1133} + F_2(\lambda_3, \lambda_2, \lambda_1)\tilde{W}_{1122},$$

$$K_{||} = \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{9\lambda_1^2}\tilde{W}_{1111},$$

$$K_{\perp} = G_1(\lambda_1, \lambda_2, \lambda_3)\tilde{W}_{2222} + G_1(\lambda_1, \lambda_3, \lambda_2)\tilde{W}_{3333} + G_2(\lambda_1, \lambda_2, \lambda_3)\tilde{W}_{2233},$$

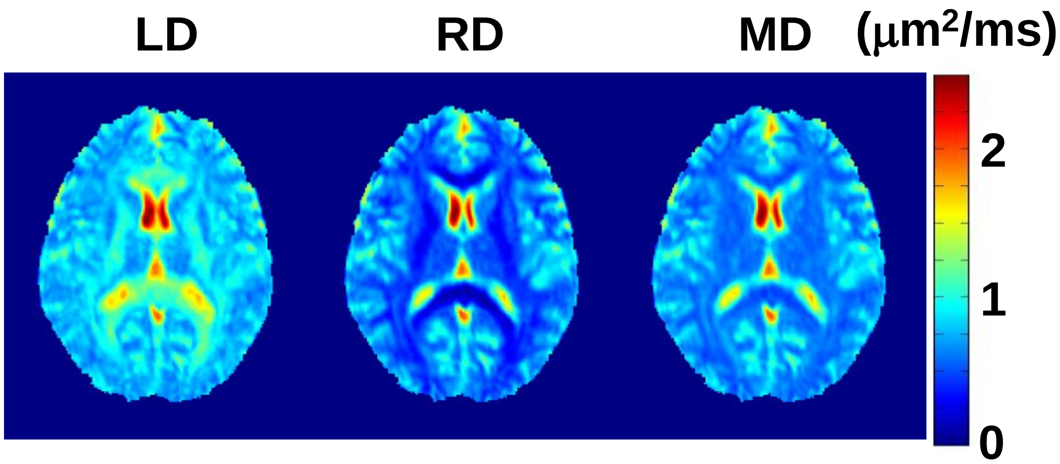
$$G_1(\lambda_1, \lambda_2, \lambda_3) = \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{18\lambda_2(\lambda_2 - \lambda_3)^2}\left(2\lambda_2 + \frac{\lambda_3^2 - 3\lambda_2\lambda_3}{\sqrt{\lambda_2\lambda_3}}\right),$$

$$G_2(\lambda_1, \lambda_2, \lambda_3) = \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{3(\lambda_2 - \lambda_3)^2}\left(\frac{\lambda_2 + \lambda_3}{\sqrt{\lambda_2\lambda_3}} - 2\right).$$

$$F_1(\lambda_1, \lambda_2, \lambda_3) \equiv \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{18(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)}\left[\frac{\sqrt{\lambda_2\lambda_3}}{\lambda_1}R_F\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right) + \frac{3\lambda_1^2 - \lambda_1\lambda_2 - \lambda_2\lambda_3 - \lambda_1\lambda_3}{3\lambda_1\sqrt{\lambda_2\lambda_3}}R_D\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right) - 1\right],$$

and

$$F_2(\lambda_1, \lambda_2, \lambda_3) \equiv \frac{(\lambda_1 + \lambda_2 + \lambda_3)^2}{3(\lambda_2 - \lambda_3)^2}\left[\frac{\lambda_2 + \lambda_3}{\sqrt{\lambda_2\lambda_3}}R_F\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right) + \frac{2\lambda_1 - \lambda_2 - \lambda_3}{3\sqrt{\lambda_2\lambda_3}}R_D\left(\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, 1\right) - 2\right].$$



Conventional DTI

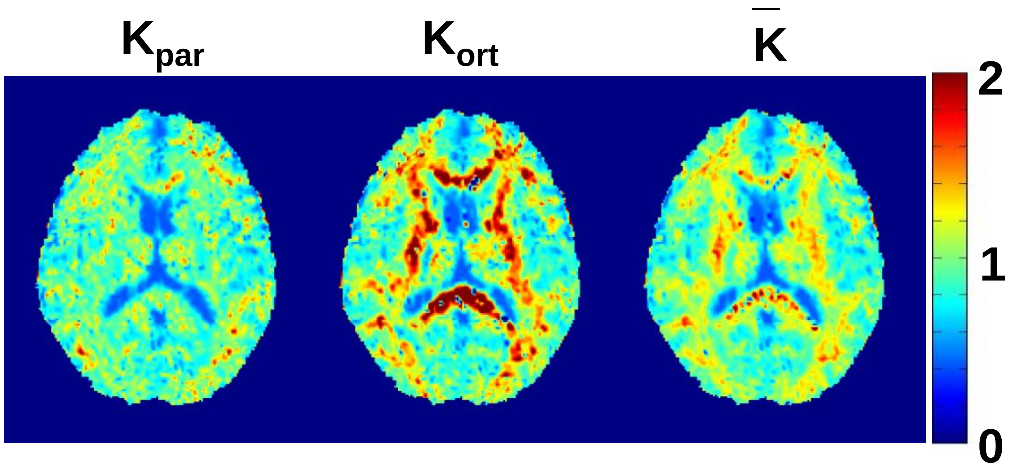
128x128x32x15

=

7.864320 10^6

linear systems to solve

~ 15 s on CPU*



Kurtosis tensor imaging

128x128x32x15

=

7.864320 10^6

non-linear fit to perform

~7200 s on CPU*

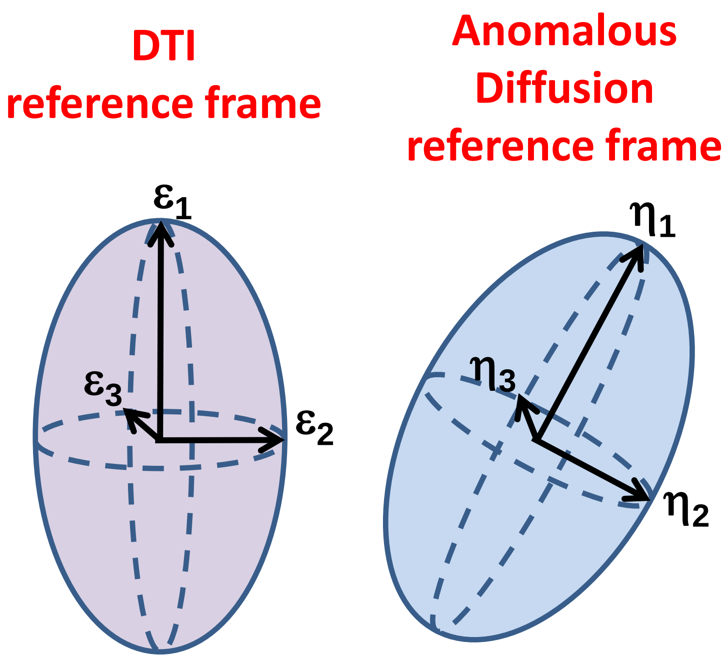
* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz

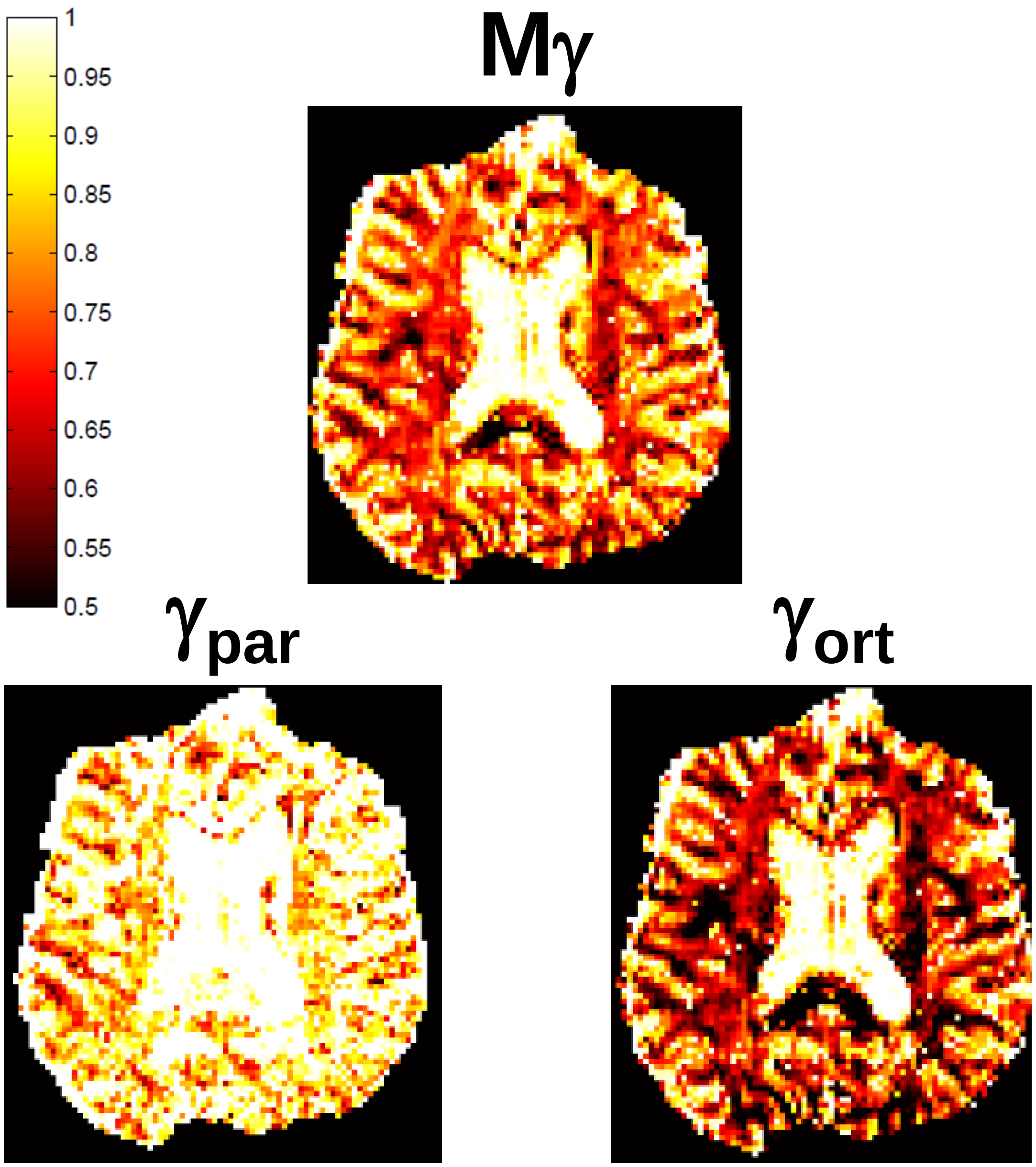
Stretched exponential model

$$S(b) = S(0)e^{-A_1(b_1^*)^{\gamma_1} - A_2(b_2^*)^{\gamma_2} - A_3(b_3^*)^{\gamma_3}}$$

$$b_i^* = \vec{b} \cdot \vec{\eta}_i$$

Anomalous diffusion reference frame





Conventional DTI

128x128x32x15

=

7.864320 10^6

linear systems to solve

~ 15 s on CPU*

Stretched exponential imaging

128x128x32x15

=

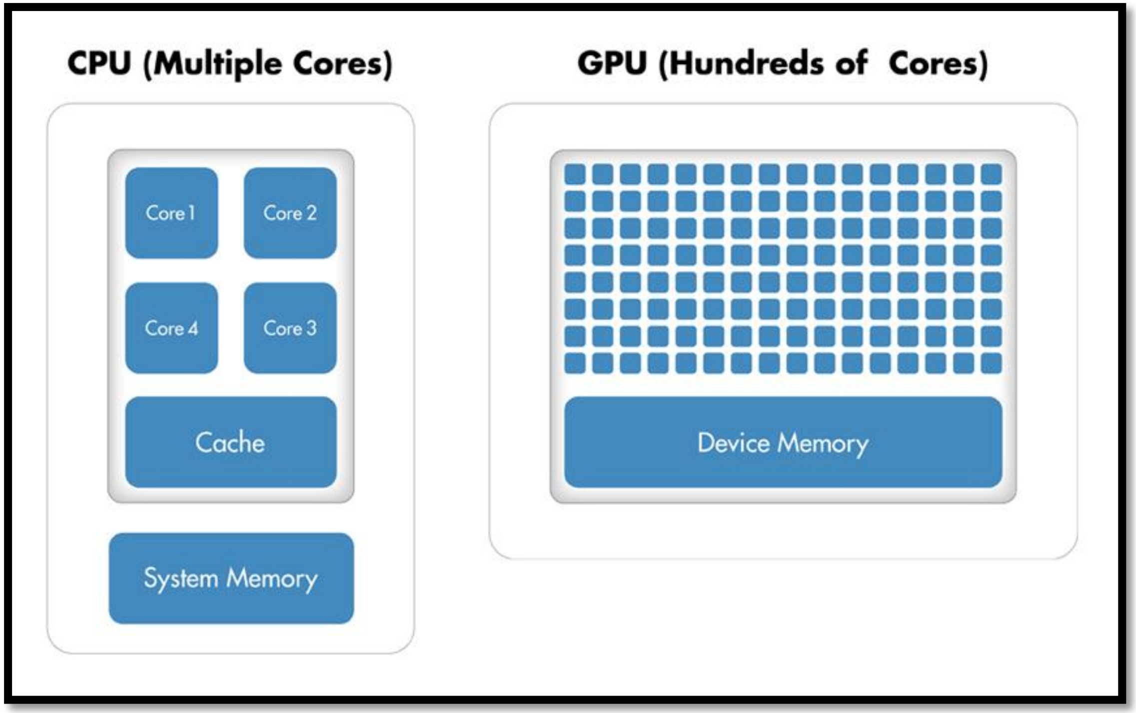
7.864320 10^6

non-linear fit to perform

~6600 s on CPU*

* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz

When GPU can accelerate an application



- **Computationally intensive** — The time spent on computation significantly exceeds the time spent on transferring data to and from GPU memory.

- **Massively parallel** — The computations can be broken down into hundreds or thousands of independent units of work.

Computationally intensive

Kurtosis tensor imaging
non-linear fit
~7 ms per pixel
vs
~ 10-80 ns to read data from memory

Stretched exponential imaging
non-linear fit
~6 ms per pixel
vs
~10-80 ns to read data from memory

Massively parallel

Kurtosis tensor imaging
~10^6-10^7 **independent** non-linear fit

Stretched exponential imaging
10^6-10^7 **independent** non-linear fit

Exemple: GPU acceleration of the solution of a wave equation in 2D*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Solution in space

Chebyshev spectral method

+

Solution in time

Second-order central finite difference method (leap-frog method)

2048 x 2048 grid *x 50* iterations ~ 80 s on CPU$

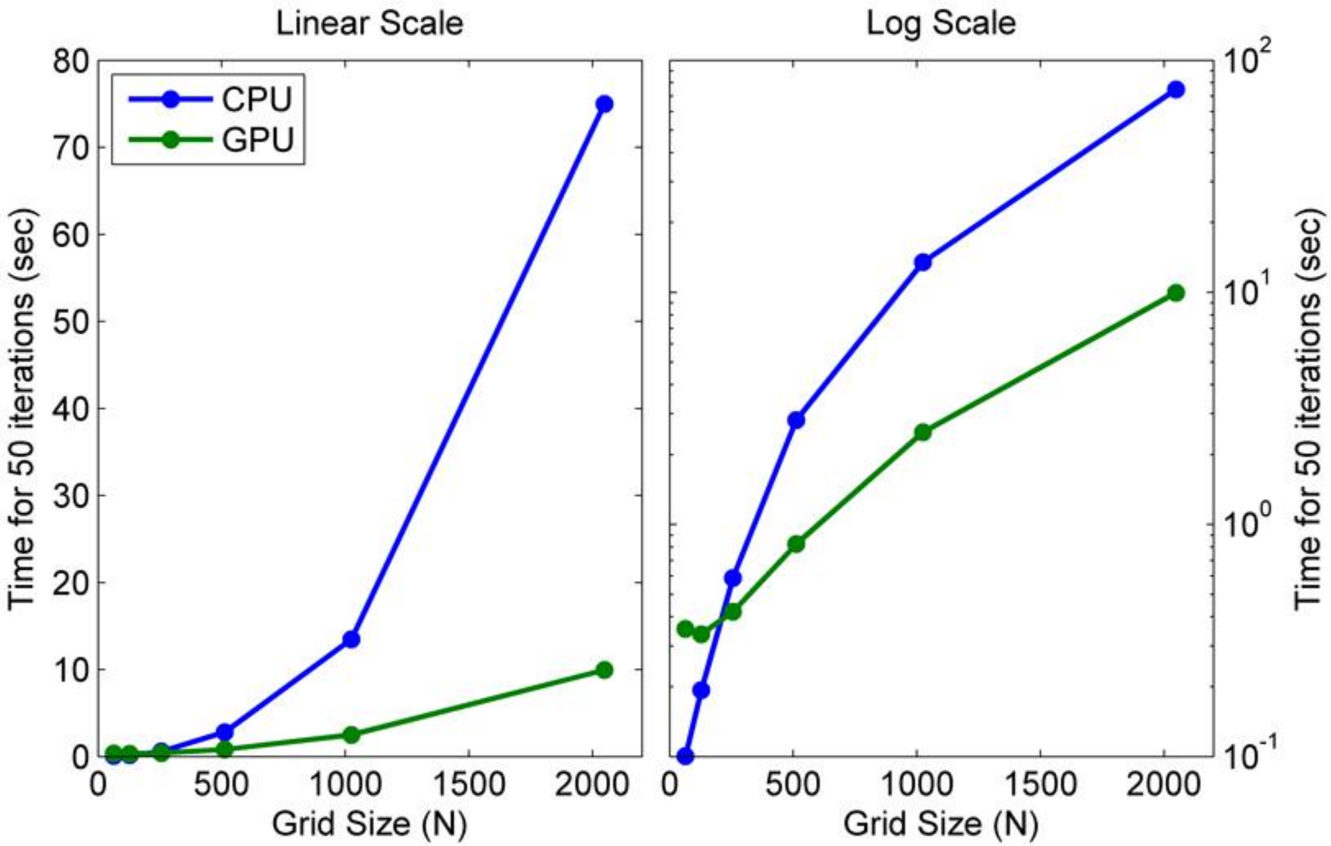
* from:
http://www.mathworks.it/company/newsletters/articles/gpu-programming-in-matlab.html

$ 12 thread on an Intel Xeon X5650 CPU at 2.66 GHz

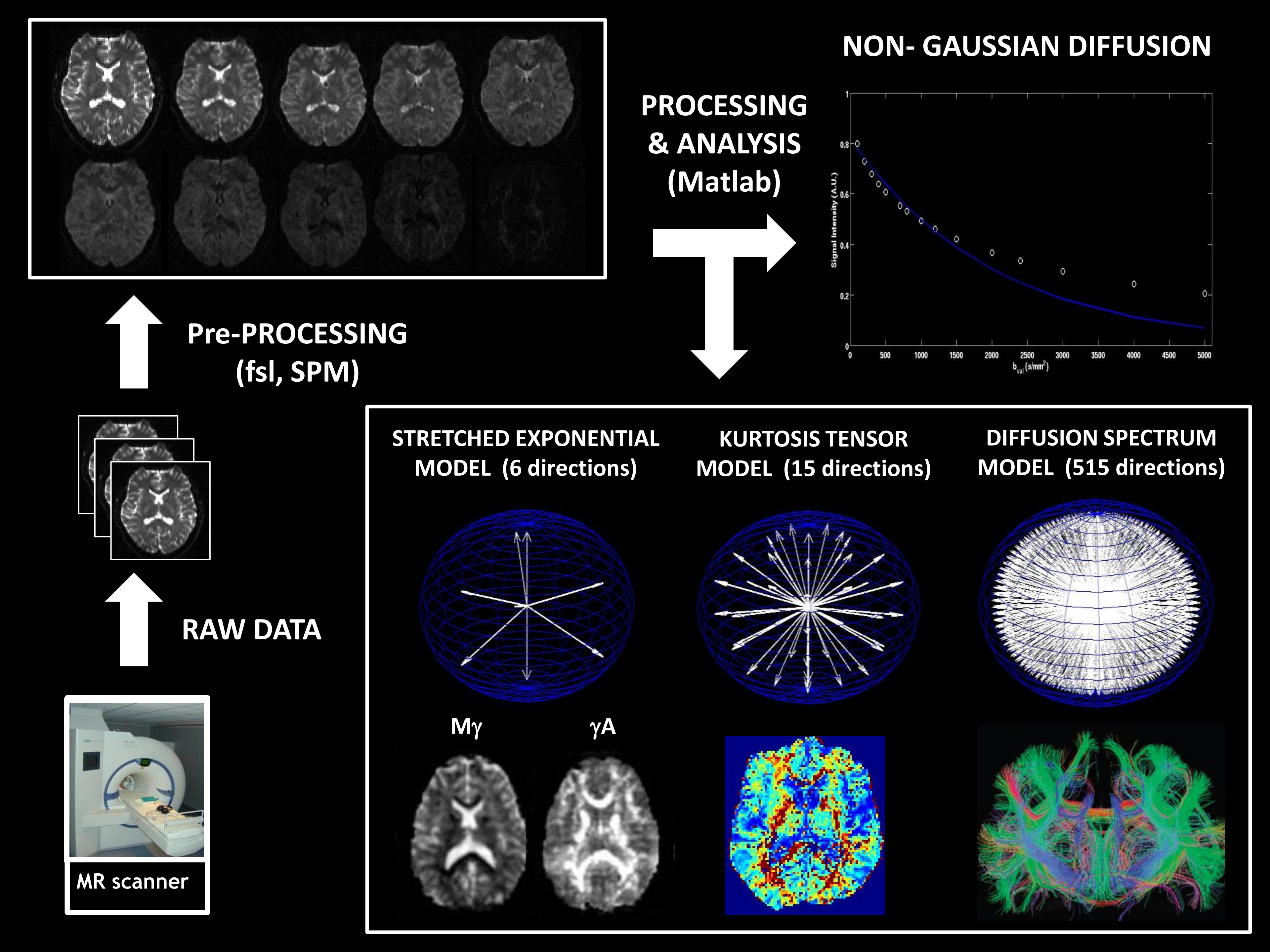
By using GPU acceleration*

7.5x decrease in compute time

2048 x 2048 grid *x 50* iterations ~ 10 s on GPU$



* from:
http://www.mathworks.it/company/newsletters/articles/gpu-programming-in-matlab.html
$ NVIDIA Tesla C2050 GPU with 448 CUDA cores at 1.15 GHz


NON- GAUSSIAN DIFFUSION
PROCESSING
& ANALYSIS
(Matlab)
Signal Intensity (A.U.)
b_{val} (s/mm^2)
Pre-PROCESSING
(fsl, SPM)
RAW DATA
MR scanner
STRETCHED EXPONENTIAL
MODEL (6 directions)
KURTOSIS TENSOR
MODEL (15 directions)
DIFFUSION SPECTRUM
MODEL (515 directions)
Mγ
γA

- Development of highly parallelized non-conventional NMR image processing algorithms;

DONE

- Non-conventional NMR image processing algorithms code optimization;

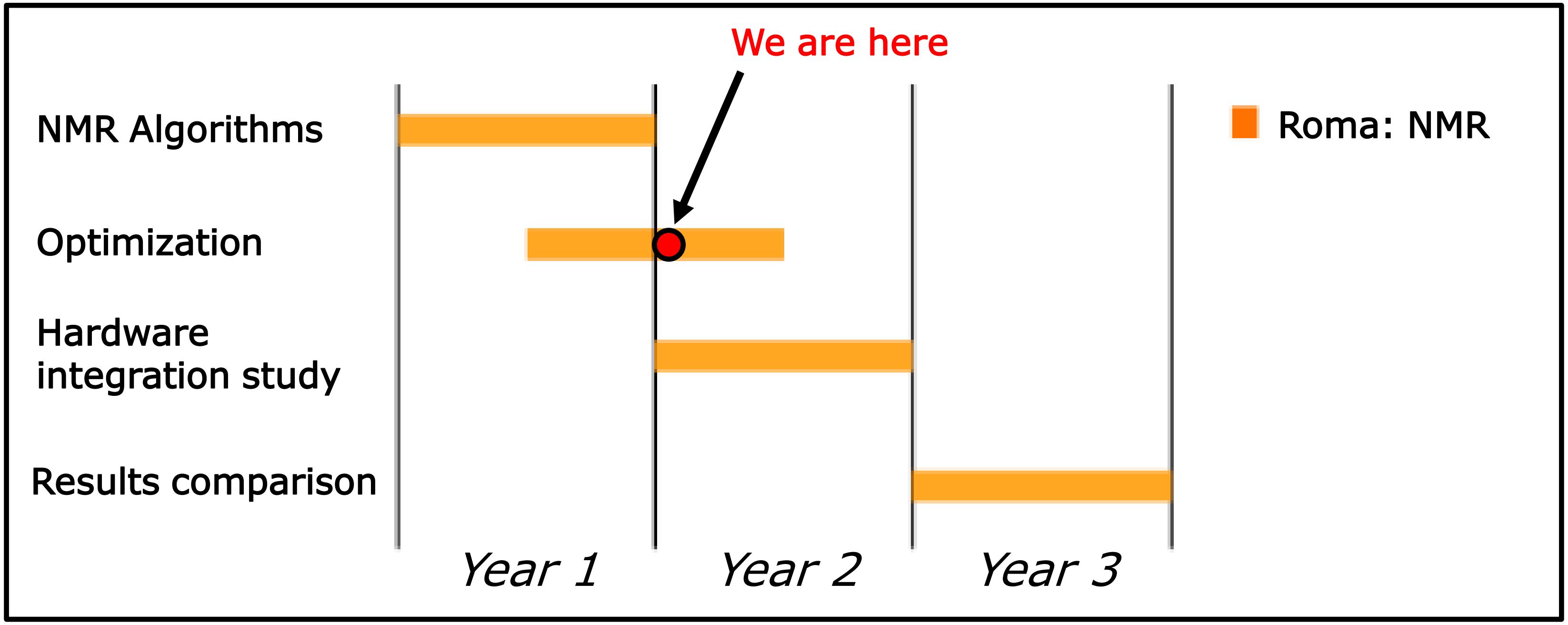
PARTIALLY DONE

- Algorithms porting on GPU and hardware integration study;

LESS THAN 1 YEAR REQUIRED

- Results comparison.

LESS THAN 1 YEAR REQUIRED



- Development of highly parallelized non-conventional NMR image processing algorithms;

DONE

- Non-conventional NMR image processing algorithms code optimization;

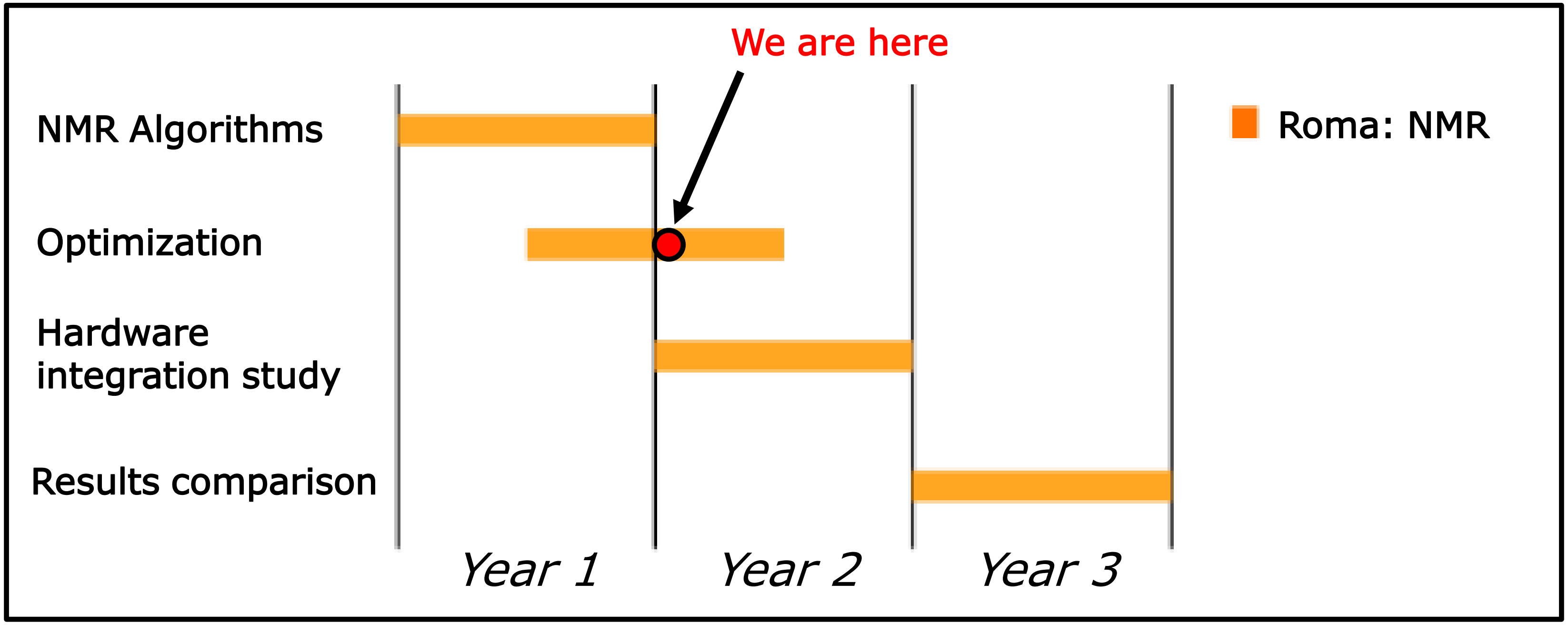
PARTIALLY DONE

- Algorithms porting on GPU and hardware integration study;

LESS THAN 1 YEAR REQUIRED

- Results comparison.

LESS THAN 1 YEAR REQUIRED



• Non-conventional NMR image processing algorithms code optimization;

Algorithms porting on GPU

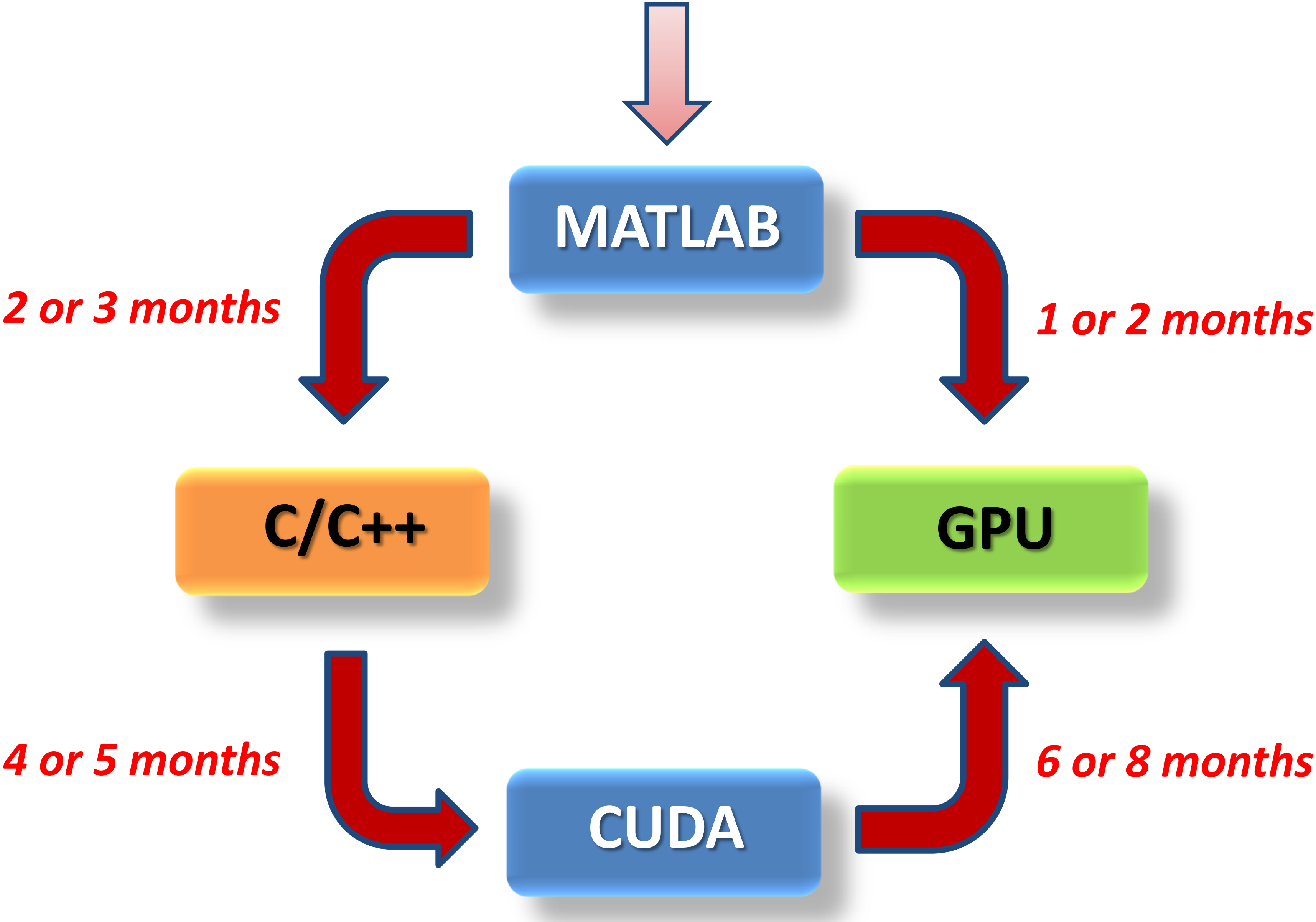
MATLAB ← *1 or 2 months* | *6 or 8 months* → **CUDA**

MATLAB

- Easier and faster (all our codes are written for Matlab)
- Bypass the study of the specific hardware architecture
- Only a single GPU can be used

CUDA

- Slower (all our codes are written for Matlab)
- Require the study of the specific hardware architecture
- Multiple GPUs can be used


MATLAB
2 or 3 months
1 or 2 months
C/C++
GPU
4 or 5 months
6 or 8 months
CUDA