GPU Accelerated QSM. *Throw your slow CPUs out of the Window(s).*

Patrick Fuchs[1], Carlos Milovic[1]
[1]University College London, Medical Physics and Biomedical Engineering

Introduction: In Quantitative Susceptibility Mapping (QSM) local magnetic susceptibility distributions are reconstructed from the phase of GRE measurements[1]. Since this relation is inherently three-dimensional this creates a computationally expensive problem. Novel optimisation algorithms try to reduce the time needed for accurate QSM reconstructions, but these often require parameter fine-tuning which may take from several minutes to a few hours. With the advent of consumer grade high performance graphical processing units (GPU) with sufficient memory to contain the three-dimensional optimization problem, leveraging this has become more feasible. In this abstract we demonstrate the processing power of a GPU by speeding up the FANSI toolbox[2].

Methods: MATLAB makes it surprisingly easy to implement GPU computing power using the *gpuArray* command. To illustrate the power and speed up this provides for (almost) no extra coding we have added GPU acceleration to the open-source FANSI toolbox and compared the performance of two different QSM algorithms on CPU and GPU. These are the ADMM-based *nlTV*[3] and the gradient descent-based *NDI*[4,5] algorithms. The algorithms differ in the number of FFTs and matrix vector products. They are run for a fixed number of iterations (*nlTV*: 100, *NDI*: 1000), on a numerical phantom, with increasing matrix sizes (from $96 \times 96 \times 32$ to $256 \times 256 \times 192$, zero-padded) to increase the computational complexity. The comparison is performed on three different computers, a laptop (MSI GF75 Thin 95C) and two desktops (two Dell Precision 5820s with different graphics hardware).

System	MSI GF75 Thin 95C	Dell Precision 5820 #1	Dell Precision 5820 #2
CPU	Intel® Core™ i7-9750H	Intel® Core™ i9-10920X	Intel® Core™ i9-10920X
RAM	32GB DDR4 – 2133MHz	64GB DDR4 - 2666MHz	64GB DDR4 - 2666MHz
GPU	NVIDIA® GeForce® GTX 1650	NVIDIA® Quadro® RTX 4000	NVIDIA® Quadro® P2000
GPU Mem	4GB	8GB	5GB
MATLAB™	2018a	2020b	2019b
OS	Fedora 32	Windows 11, Ubuntu 20.04	Kubuntu 18.04

Table 1. A table comparing the hardware of the different systems.

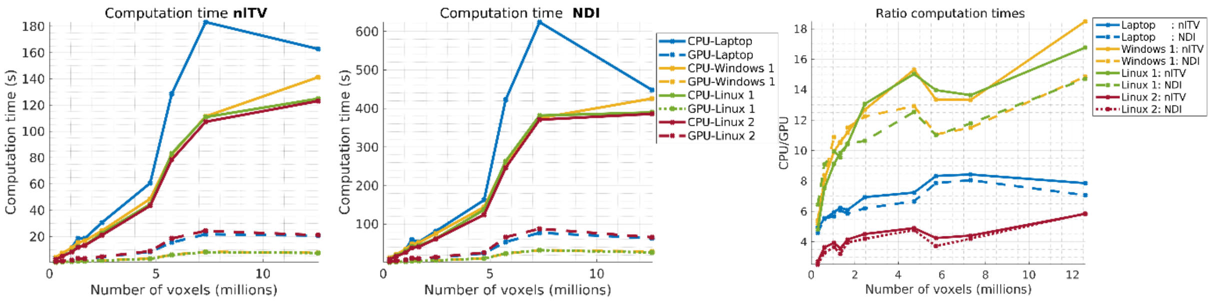

Figure 1. Computational times for the *nlTV* (left) and *NDI* (middle) algorithms. Solid lines are CPU times and dashed lines are GPU times. Ratio between CPU and GPU computation times(right). The value denotes the amount of speedup achieved by running the algorithms on the GPU as opposed to the CPU. Solid lines denote *nlTV* and dashed lines the *NDI* algorithm

Results: The computation times for *nlTV* and *NDI* can be found in Figures 1 (left and middle). A comparison between the acceleration factors is presented in Figure 2 (right).

Discussion: Both algorithms show significant speedup on the GPU compared to the CPU implementation, the *nlTV* algorithm benefits slightly more, probably due to its larger reliance on FFTs and lower iteration count. Furthermore, it was found that both CPU and GPU based implementations (typically) run faster on a Linux based operating system than a Windows based one, possibly because of the memory and computational overhead and management of the operating system. The observed drop in speedup is most likely to the efficiency of the FFTW algorithm for matrix sizes factors of 2.
The limited memory available is a significant bottleneck to high resolution QSM reconstructions, as a matrix size of $512 \times 512 \times 512$ would be too large for any of the graphics cards tested here. Lastly, unfortunately GPU computation in MATLAB is constrained to NVIDIA hardware (as is the case for most high-level programming language) since the support is through the CUDA language.

Conclusion: Using even a four-year-old GPU significantly outperforms current state of the art CPU processors on large enough QSM reconstruction problems, for which GPUs provide between a 2.5 up to 18 times. We hope this inspires others to speed up their open-source tools using the GPU.

References: [1]Schmueli, K., Academic Press, 2020(1):819-838. [2]FANSI toolbox: https://gitlab.com/cmilovic/FANSI-toolbox/. [3]Milovic, C. et al., Magn Res Med. 2018, 4Polak, D, et al., NMR Biomed. 2020. [5]Milovic, C. et al., ISMRM 2021.