CATEGORY: ASTRONOMY & ASTROPHYSICS - AA05

POSTER P5316 | CONTACT NAME Tim McMahon: mcmahon@milkyway.gsfc.nasa.gov

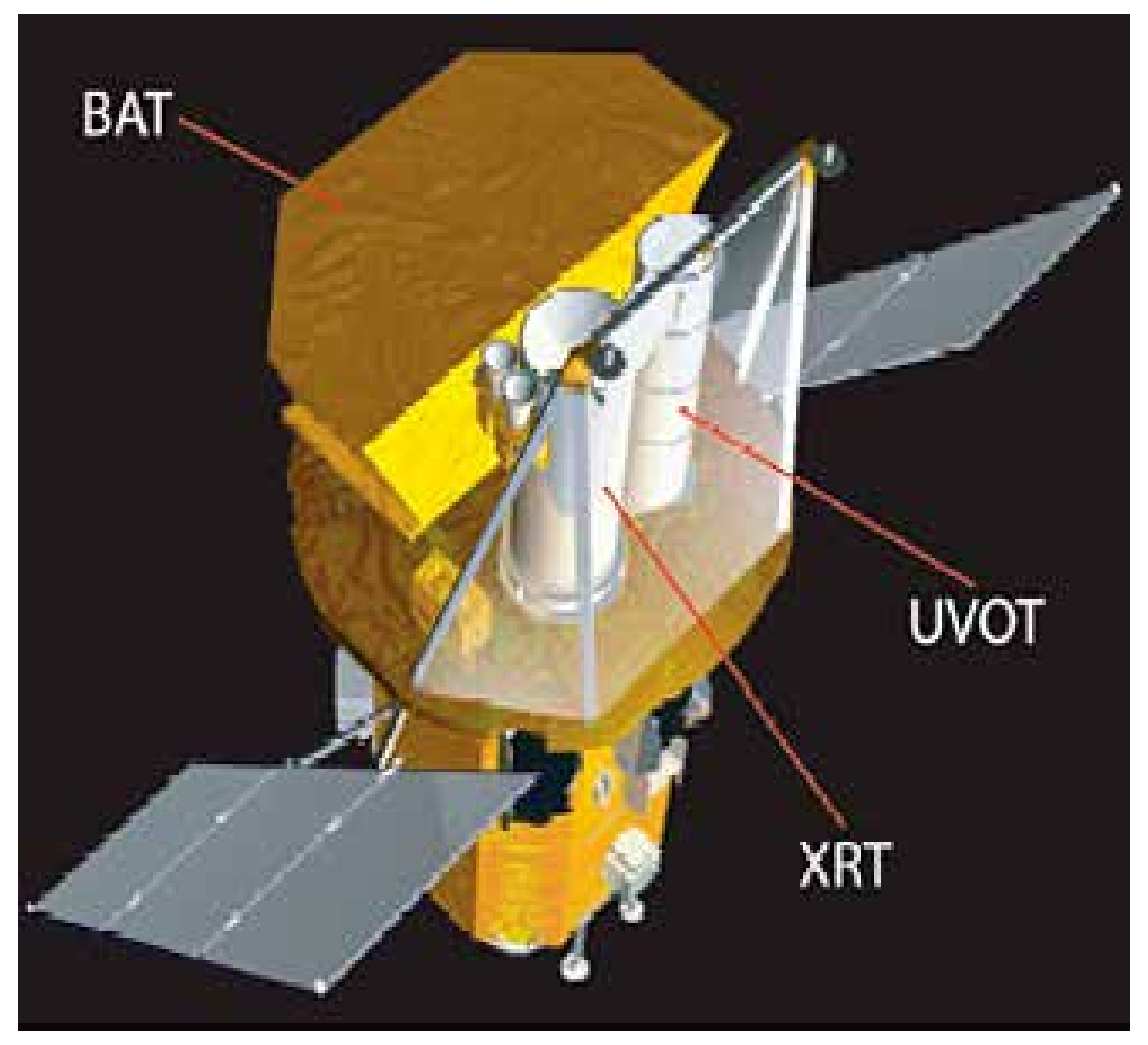
GPU TECHNOLOGY CONFERENCE

Astrophysical Gamma-Ray Source Imaging with NASA's Swift Telescope using Nvidia GPUs

Tim McMahon, Langston University

NASA's Swift Telescope

The Swift satellite is designed to autonomously slew to gamma-ray burst (GRB) events and when not triggered for GRBs will collect data in survey mode obtaining long exposure gamma-ray images of the sky with the Burst Alert Telescope (BAT). The survey images from the BAT telescope are used to study the high-energy dynamics of accretion disks and jet production around supermassive blackholes of Active Galactic Nuclei.



NASA

NASA

The BAT telescope is a wide field of view telescope covering 1/8 of the sky and detects photons with energies from 15 keV to 150 keV. To image gamma-ray sources the BAT telescope has a coded aperture mask with 50,000 square lead tiles in pseudo-random pattern and a 32,768 element CZT detector array positioned 1 meter below the mask. Astronomical gamma-ray point sources will cast a shadow of the mask onto the detector plane with an offset dependent upon the angular separation of the point source relative to the axis of the BAT telescope. Multiple sky gamma-ray point sources will give a summed CZT detector array count that comes from multiple offset mask shadow photons. A deconvolution algorithm is used to reconstruct the sky points back from the summed CZT detector counts.

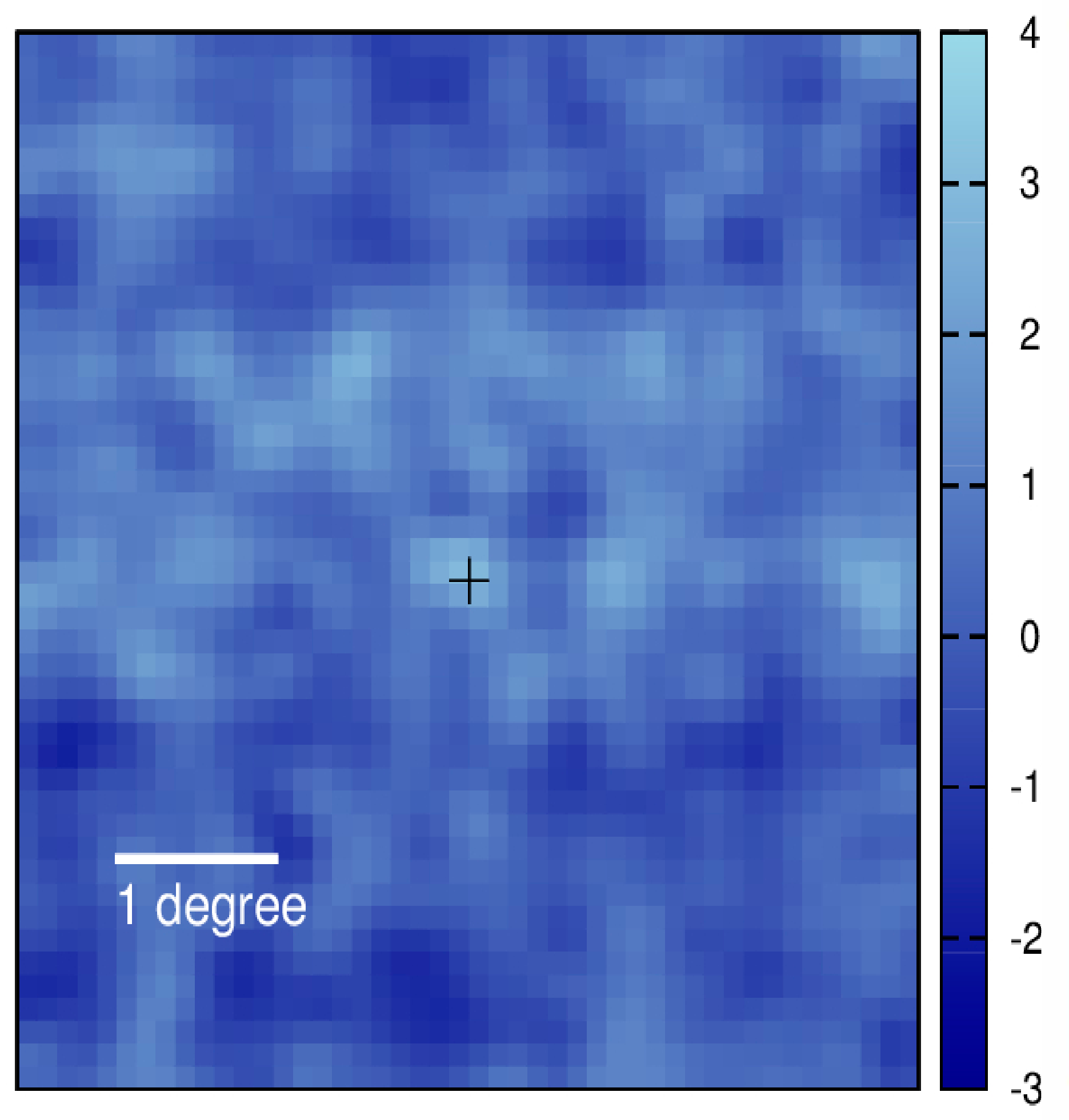
Detector and Imaging Algorithms

Typically for high energy detectors a Monte Carlo approach is used to simulate the response of the detector for ionizing radiation. For the BAT detector, however, mass absorption of gamma-rays is the dominant physical process. We found that instead of a Monte Carlo integration the transmission and absorption of gamma-ray photons could be mathematically integrated for the mask and CZT volumes. For the integral we define an elemental volume called a tavm which is the volume swept out by a polygon transported between two planes that have any relative orientation. Any collection of polyhedral volumes can be decomposed into axis parallel tavms. Photons from a sky point source will be parallel to each other and the flux volume through a polygonal surface of a mask or CZT volume will be a tavm. One photon can traverse multiple instrument volumes and will generate a tavm string that can be integrated once and calculate the CZT photon absorption as a function of incident photon energy.

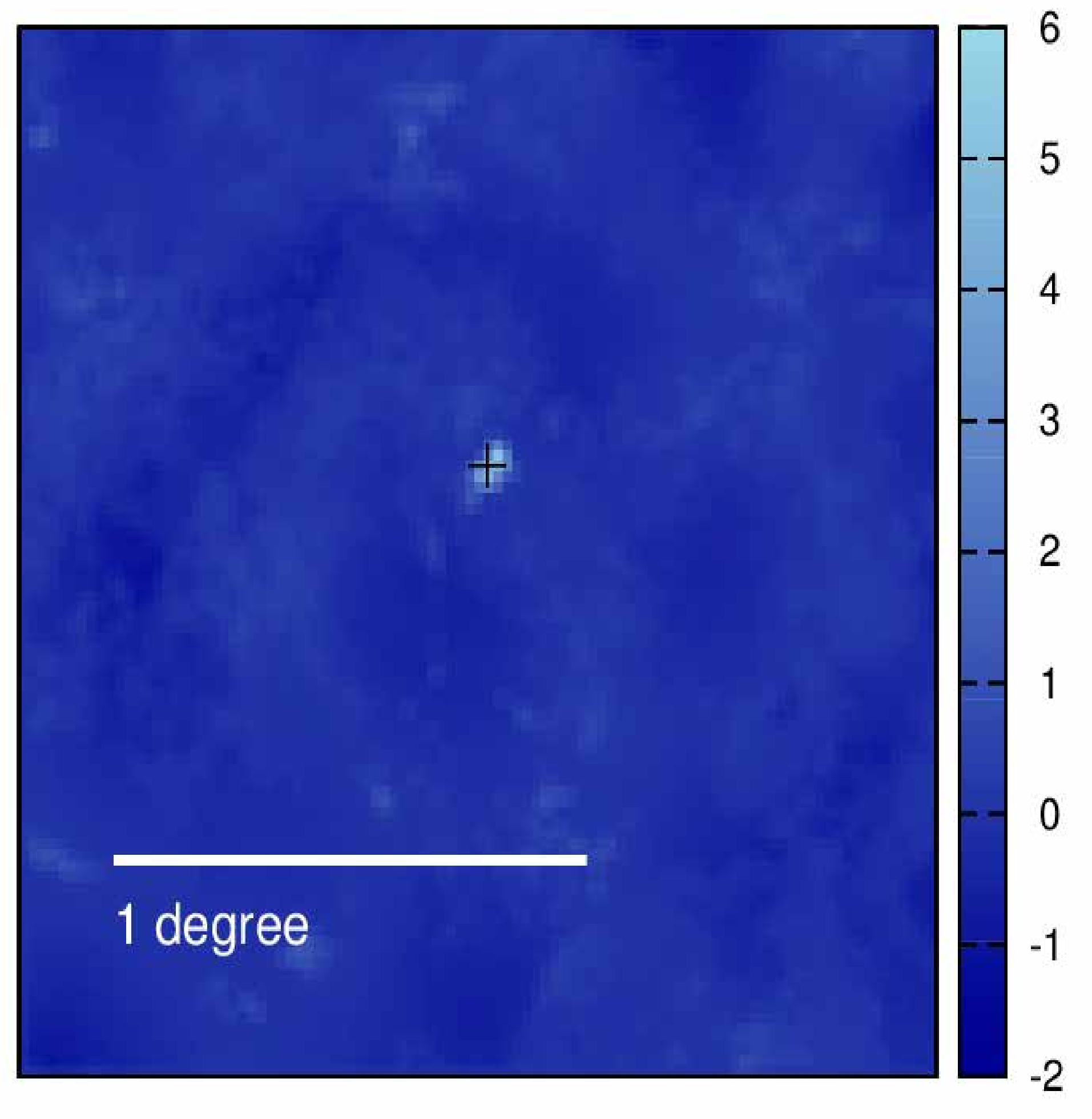
The raw point spread function of the BAT coded aperture telescope is approximately the CZT detector pitch. We have developed an image reconstruction algorithm which reprocesses a BAT image assuming point sources to improve the point spread function and reduce background contributions.

Displayed in the figures are the reprocessing of a BAT image with the algorithms and yields an approximate four-fold increase in signal to noise and an attendant reduction in the point spread function. Each BAT image consists of 2 million pixels and to date over 150,000 survey exposures have been archived by Swift.

These algorithms of modeling detector volumes, deconvolving detector plane images to produce BAT images, and the reprocessing of BAT images to produce higher S/N ratio images are amenable to parallelization on GPU processors. In all three cases the inter-thread communication is minimal since the detector volumes and pixels are independent of each other. In addition the ratio of kernel computation load versus host data transfer is high which makes implementing the algorithms less susceptible to PCI express data transfer rates between the host CPU and Tesla K20 GPU.



BAT image NGC 4151



BAT image NGC 4151 reprocessed

GPU Implementation: experience converting a large software package to be GPU enabled

The BAT detector modeling and image reconstruction software was originally written without any design for GPU processing. Creating a GPU enabled version of the more than 20,000 lines of code and hundreds of functions was expected to be done with a minimal amount of programming hours while at the same time achieving optimal compute performance. The GPU version also was required to have the same functionality and be error free. A GPU with double precision capabilities like the Tesla K20 was also a requirement.

To achieve these goals we adopted a staged approach of redesigning the code to have the framework of a host to GPU device design that would run initially only on the host cpu. Functions that would eventually run as device kernel code were identified and all references to globally defined variables removed. All data transfers between functions identified to run on either host or gpu were reworked to funnel through one gpu wrapper function. At each stage of the code rewriting the software processed a standard image file to identify if it was numerically equivalent. In this way a functionally equivalent version of the software package with a framework of host-gpu architecture that ran on host only was made. The last stage was to create device kernel mirror functions running under CUDA 6.5 of the gpu identified functions to create a GPU enabled version of the original software package. Approximately 10% of the resulting software consists of GPU kernel code.

Acknowledgements

I would like to thank Nvidia for providing a Tesla K20 board for use on this project. Part of this work was performed as a Cycle-10 Swift Guest Investigator under NASA grant number NNX14AJ05G.