Projected Conjugate Gradient Solvers on GPU and Its Applications

Youzuo Lin youzuo.lin@asu.edu, Rosemary Renaut renaut@asu.edu, Arizona State University

School of Mathematical & Statistical Sciences

Contribution

- A linear system with multiple right hand sides is proposed to model the 3-D image reconstruction application.
- Information shared from consecutive right hand sides can be utilized to reduce the number of matrix vector multiplication (Mat-Vec) on GPU.
- We notice that BLAS 3 outperforms BLAS 1 and 2, which is utilized to further optimize the reconstruction algorithm.

Problem Description and Modeling

- A 3-D image reconstruction can be modeled as

$$A\mathbf{x} = B = [\mathbf{b}^{(1)}, \dots \mathbf{b}^{(s)}], \tag{1}$$

where A is the reconstruction matrix, $\mathbf{b}$ is the measurement gathered from the medical device.

- Because of the ill-posedness of the reconstruction matrix, regularization (Tikhonov) needs to be utilized:

$$\arg\min_{\mathbf{x}} \left\{ \frac{1}{2}\|A\mathbf{x} - B\|_2^2 + \frac{\lambda}{2}\|L\mathbf{x}\|_2^2 \right\}. \tag{2}$$

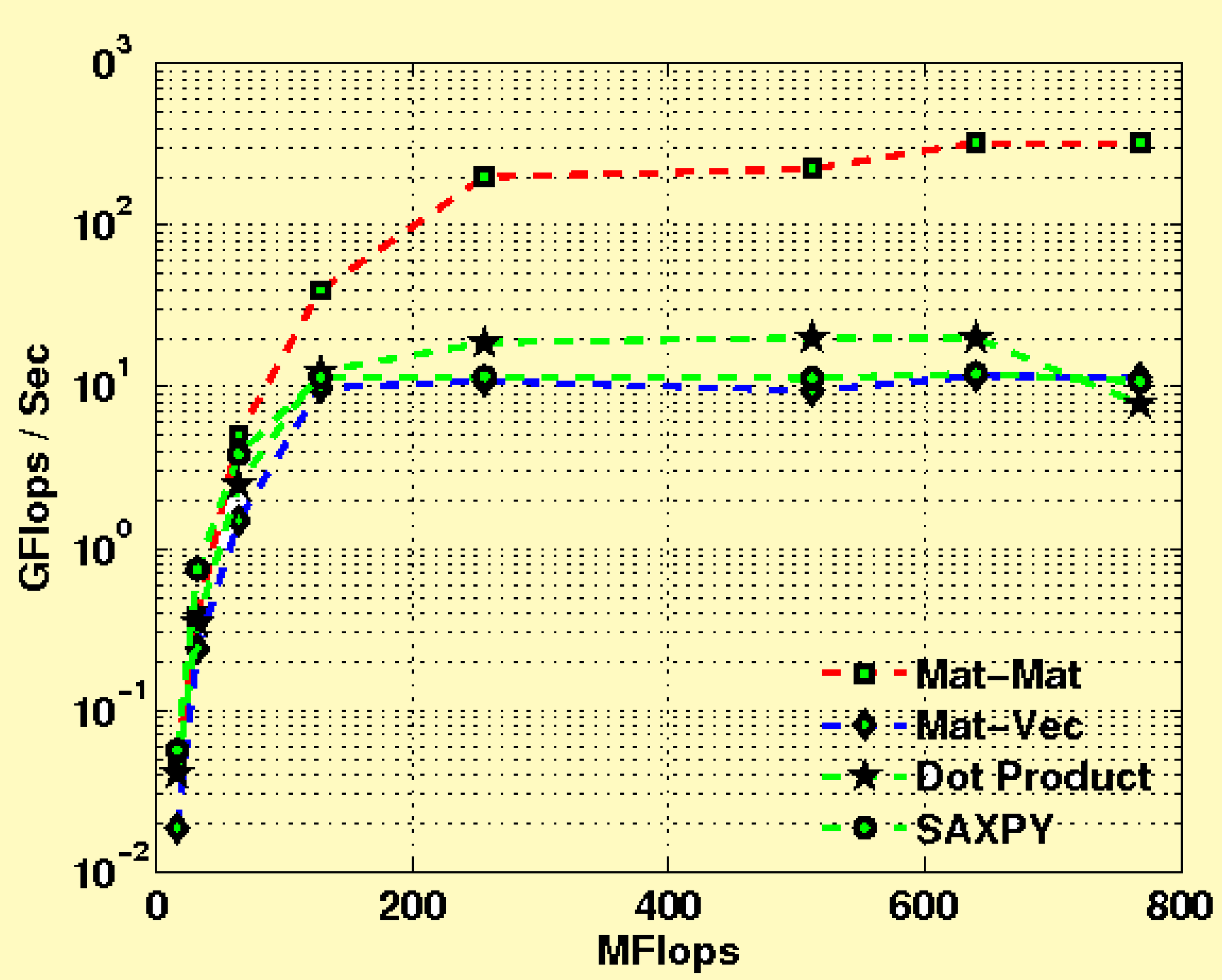
This is equivalent to solving a linear systems:

$$(A^T A + \lambda^2 L^T L)\mathbf{x} = A^T B. \tag{3}$$

- The conjugate gradient (**CG**) solvers is a standard solver used for solving the linear system (3).

BLAS Performance on GPU

- The performance of the GPU (Nvidia Tesla C1060/CUDA+Cublas) in **FLOPS** is compared for the BLAS 1, 2 and 3 kernels.
- All the kernels will be set up to be the same FLOP count of $2n^3$.



- **Conclusion:** It is interesting to see that **BLAS 3 kernels definitely outperform both BLAS 1 or BLAS 2 kernels in terms of FLOPS**.

Implementation on GPU

- **Computation Bottleneck**: The CG algorithm is heavily relies on the Mat-Vec.
- **Optimization One**: To reduce the number of Mat-Vec's.
- **Optimization Two**: To take advantage of the better performance of BLAS 3 over both BLAS 1 and 2.

Optimization One - Projected CG (PrCG)

- **Idea**: Galerkin projection can be used to project the current system to the previous generated solution space in CG algorithm [1].

Input: TOL, $i = 0$, RelRes $= 1$, k
Output: $\mathbf{x}_q^{(k)}$
if It is the seed system **then**
  Canonical CG();
else
  $\mathbf{r}_q^{(i)} = \mathbf{b}_q^{(i)} - A\mathbf{x}_q^{(i)}$
  for $i = 1$ to k **do**
    $\alpha_q^{(i)} = \dfrac{< \mathbf{p}_1^{(i)}, \mathbf{r}_q^{(i)} >}{< \mathbf{p}_1^{(i)}, A\mathbf{p}_1^{(i)} >}$
    $\mathbf{x}_q^{(i)} = \mathbf{x}_q^{(i)} + \alpha_q^{(i)}\mathbf{p}_1^{(i)}$
    $\mathbf{r}_q^{(i+1)} = \mathbf{r}_q^{(i)} - \alpha_q^{(i)}A\mathbf{p}_1^{(i)}$
  end for
  RelRes $= \|\mathbf{r}_q^{(i)}\|_2 / \|\mathbf{b}_q\|_2$
end if
Restart if further refinement needed

Optimization Two - Augmented PrCG (APrCG)

- **Idea**: A further speedup can be reached if a mathematically equivalent algorithm can be expressed in terms of BLAS 3 operations other than BLAS 1 or 2 operations.

Utilizing the orthogonality relationships that (detailed derivation is omitted because of the page limitation),

$$< \mathbf{p}_1^{(j)}, A\mathbf{p}_1^{(i)} > = 0, \quad j \neq i, \tag{4}$$

and

$$< \mathbf{p}_1^{(j)}, \mathbf{r}_q^{(i)} > = 0, \quad j < i, \tag{5}$$

we proposed a Augmented Projected CG **(APrCG)** algorithm mathematically equivalent to the **(PrCG)**.

Input: TOL, $i = 0$, RelRes $= 1$, k
Output: $\mathbf{x}_q^{(k)}$
$\mathbf{r}_q^{(i)} = \mathbf{b}_q - A\mathbf{x}_q^{(i)}$
$\boldsymbol{\alpha} = < P, \mathbf{b}_q > ./\mathrm{diagVec}(< P, AP >)$
$\Lambda = \mathrm{diag}(\boldsymbol{\alpha})$
$\mathbf{x}_q = \mathbf{x}_q^{(0)} + \mathrm{sum}(P \cdot \Lambda)$
$\mathbf{r}_q = \mathbf{r}_q^{(0)} - \mathrm{sum}(AP \cdot \Lambda)$
RelRes $= \|\mathbf{r}_q\|_2 / \|\mathbf{b}_q\|_2$

Results - 3D Image Reconstruction

- A 3D Shepp-Logan Phantom $128 \times 128 \times 128$ is utilized for testing, system matrix size: 16650×16384, condition number 1.1×10^{32}.

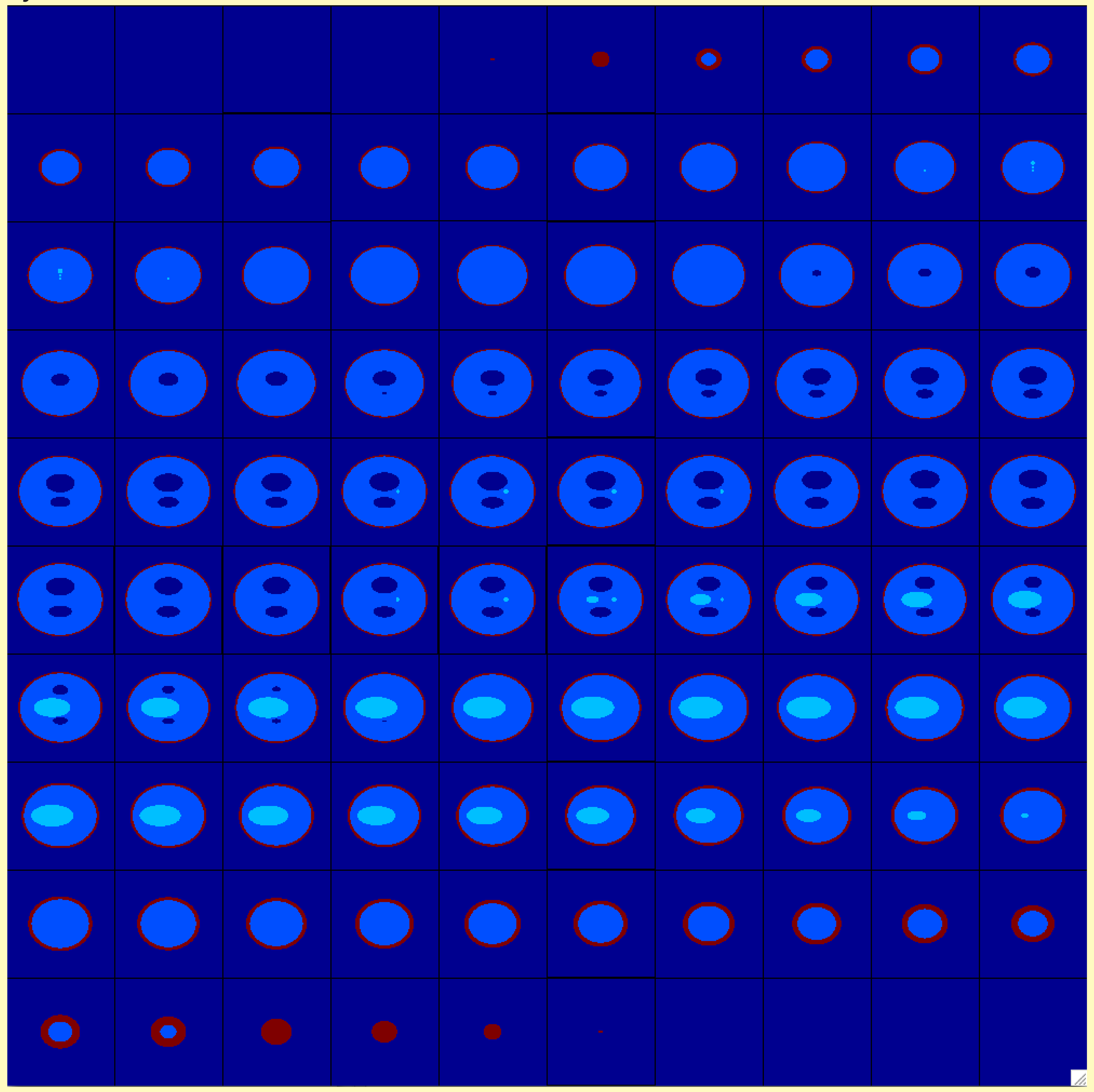


Figure: Slice Show of A 3D Shepp-Logan Phantom

Slice	CG		PrCG		APrCG		Speedup	ImpRatio
	Cost	SNR	Cost	SNR	Cost	SNR		
65	17.41	13.67	2.16	13.67	2.06	13.67	8.45	4.63%
66	16.34	13.80	2.74	13.83	2.58	13.83	6.33	5.84%
67	14.35	13.91	3.11	13.95	2.81	13.95	5.11	5.84%
68	13.03	13.80	3.53	13.83	3.46	13.83	3.77	5.84%

Table: Reconstruction of Four Consecutive Slices from 65 to 68 on the GPU (Nvidia Tesla C1060). The 64^{th} slice is selected as seed. **Speedup** is the speedup ratio of **APrCG** over **CG**, **ImpRatio** is the improvement ratio of **APrCG** over **PrCG**.

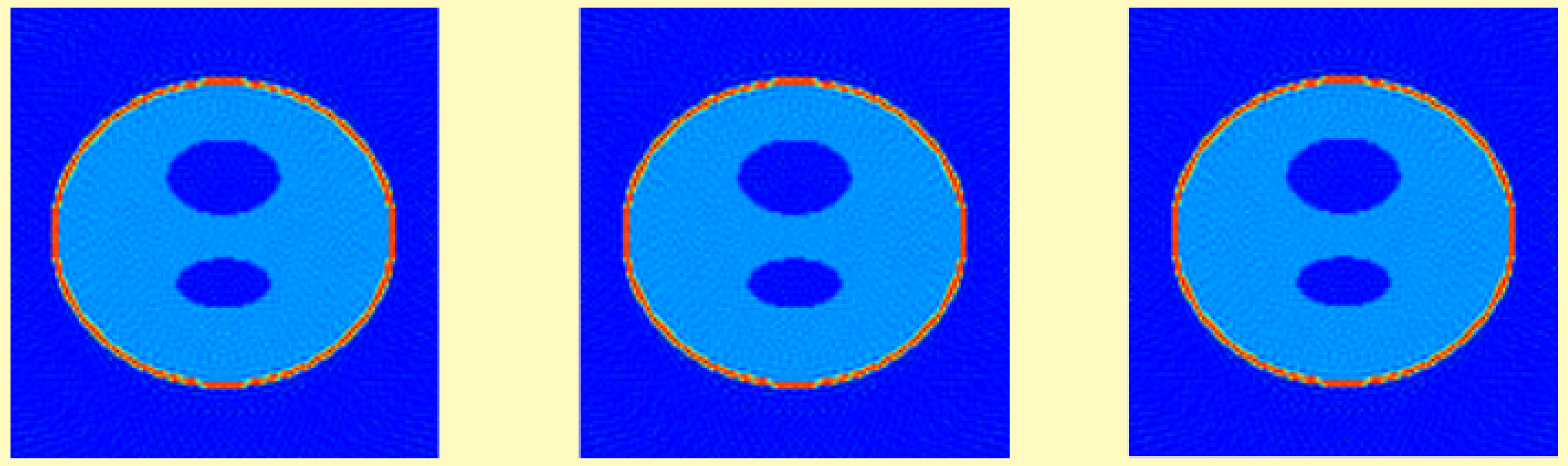


Figure: Reconstruction Results on CPU and GPU, slice index = 66. **Left**: CPU/CG; **Middle**: GPU/PrCG; **Right**: GPU/APrCG.

Reference

[1] T. F. Chan, W. L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides In *SIAM: SISC, 1997.*