Experimenting Iterative Methods for Inverse Problems at Low Precision Levels

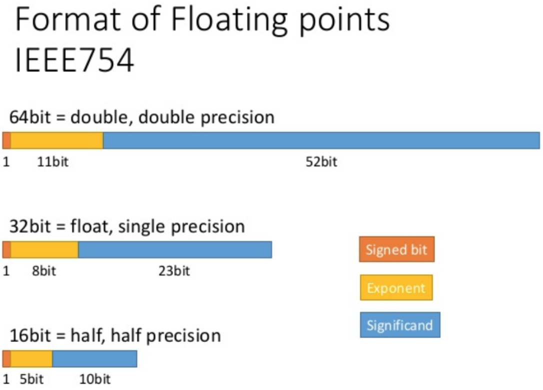
Riley Chen, Kristina Gong, Zoe Ji
Emory Summer REU 2022
Emory University
Atlanta, GA, USA

Outline

1. Chop
2. CGLS
3. Experiment

Chop: Overview

A closer look at double, single, fp16 precision:



[1]

[1] *Using tensor cores for mixed-precision scientific computing*. Oct-2021. URL: https://developer.nvidia.com/blog/tensor-cores-mixed-precision-scientific-computing/

Chop: Overview

- Simulate low-precision arithmetics
- Need to chop each operation

```
options.format = 'fp16';
chop([],options);

x = chop(x);
y = chop(y);
z = chop(z);
s = chop(x + chop(y * z));
```

```
options.format = 'c';
options.params = [11,23]
chop([],options);
```

Chop: Blocking

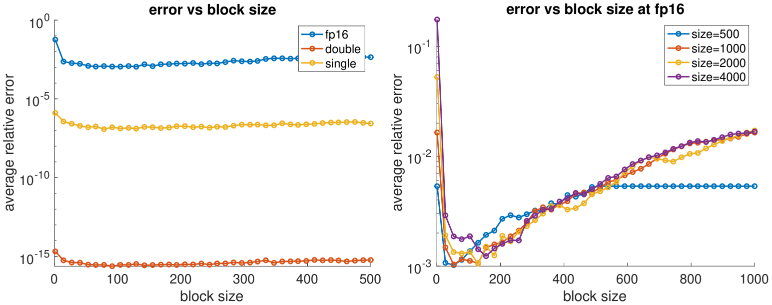
- Break an inner product into several smaller inner products
- Compute them independently and then sum

```
x = [x1, x2, x3, x4, x5, x6]
y = [y1, y2, y3, y4, y5, y6]

x_1 = [x1, x2, x3]
y_1 = [y1, y2, y3]

x_2 = [x4, x5, x6]
y_2 = [y4, y5, y6]
```

Chop: Blocking



Chop in Action

Matrix-vector multiplication:

```
function C = mv_blocked(X,y,block_size)
%
% compute the product of a matrix and a vector with chop
%

A = chop(chop(X).*chop(y'));

if nargin < 3
    block_size = 256; % default block size
end

[m, n] = size(X);

k = floor(n/block_size);
C = zeros(m,1);

for i = 1:k
    a=zeros(m,1);
    for j = (i-1)*block_size+1 : i*block_size
        a = chop(a+A(:,j));
    end
    C = chop(C + a);
end

if n-k*block_size ~= 0
    b=zeros(m,1);
    for i = k*block_size+1:n
        b = chop(b+ A(:,i));
    end
    C = chop(C + b);
end
```

```
Instead of:
a = chop(a+chop(chop(X(:,j))*chop(y(j))));
```

CGLS: Overview

- Conjugate Gradient Method: Solve $Ax = b$ for SPD matrices
- CGLS:
  - Generalize to all the matrices
  - A $\rightarrow$A^{T}A, b $\rightarrow$A^{T}b without explicitly calculating A^TA

CGLS: Our Modification

Chop each operation! :)

ALGORITHM 7.4.1. CGLS. Let $x^{(0)}$ be an initial approximation, set

$$r^{(0)} = b - Ax^{(0)}, \quad p^{(0)} = s^{(0)} = A^T r^{(0)}, \quad \gamma_0 = \|s^{(0)}\|_2^2,$$

and for $k = 0, 1, 2, \dots$ while $\gamma_k > tol$ compute

$$\begin{aligned} q^{(k)} &= Ap^{(k)}, \\ \alpha_k &= \gamma_k / \|q^{(k)}\|_2^2, \\ x^{(k+1)} &= x^{(k)} + \alpha_k p^{(k)}, \\ r^{(k+1)} &= r^{(k)} - \alpha_k q^{(k)}, \\ s^{(k+1)} &= A^T r^{(k+1)}, \\ \gamma_{k+1} &= \|s^{(k+1)}\|_2^2, \\ \beta_k &= \gamma_{k+1}/\gamma_k, \\ p^{(k+1)} &= s^{(k+1)} + \beta_k p^{(k)}. \end{aligned}$$

```
Ax = mv_blocked(A, x);normr2 = vv_blocked(d(:),d(:))
d = mv_blocked(A', b);
d = chop(d - mv_blocked(A', Ax));
normr2 = vv_blocked(d(:),d(:));

Ad = mv_blocked(A, d);
alpha = chop(normr2/normAd2);
x  = chop(x + chop(alpha*d));
r  = chop(r - chop(alpha*Ad));
s  = mv_blocked(A', r);
q = s; normr2_new = vv_blocked(q,q);
beta = chop(normr2_new/normr2);
d = chop(s + chop(beta*d));
```

[2]

[2] Åke Björck. *Numerical methods for least squares problems.* SIAM, 1996

Experiment: Image Deblurring (no noise)

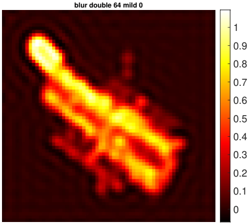


Figure: Double precision problem size 64 with mild blurring.

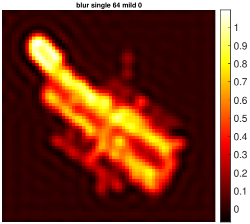


Figure: Single precision problem size 64 with mild blurring.

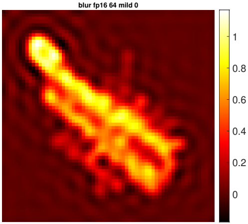


Figure: Half (fp16) precision problem size 64 with mild blurring.

Experiment: Image Deblurring (no noise)

-The error norm:

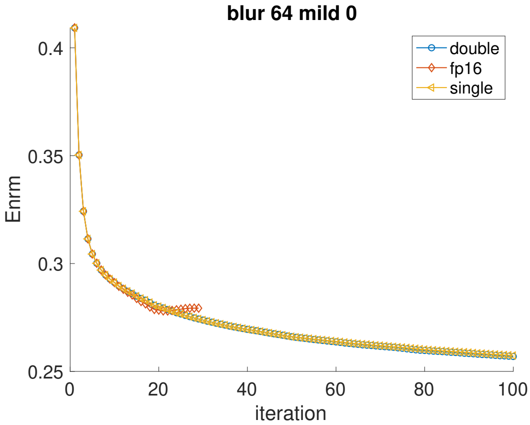


Figure: The error norm of a size 64 problem with mild blurring of different precisions.

Experiment: Image Deblurring (with noise)

-Single

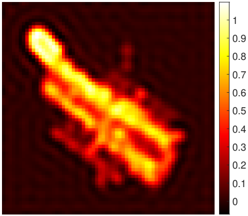


Figure: Single precision, problem size 64 with mild blurring and 0.1% noise.

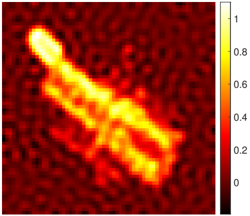


Figure: Single precision, problem size 64 with mild blurring and 1% noise.

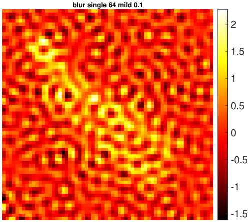


Figure: Single precision, problem size 64 with mild blurring and 10% noise.

Experiment: Image Deblurring (with noise)

-Half

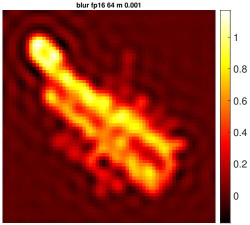


Figure: Half precision, problem size 64 with mild blurring and 0.1% noise.

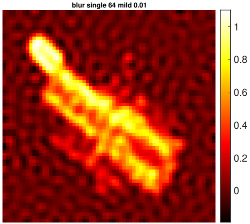


Figure: Half precision, problem size 64 with mild blurring and 1% noise.

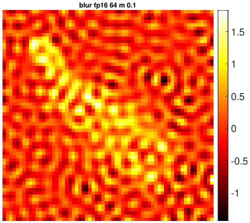


Figure: Half precision, problem size 64 with mild blurring and 10% noise.

Experiment: Image Deblurring (with noise)

- Error norm

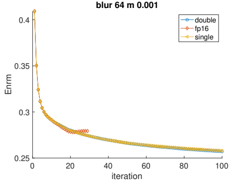


Figure: Error norm for problem size 64 with mild blurring and 0.1% noise.

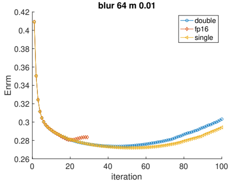


Figure: Error norm for problem size 64 with mild blurring and 1% noise.

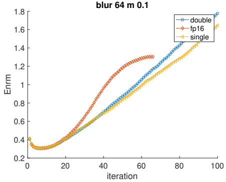


Figure: Error norm for problem size 64 with mild blurring and 10% noise.

Experiment: Tomography (no noise)

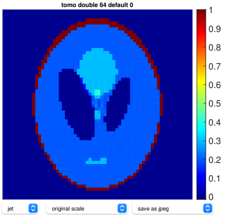


Figure: Double precision problem size 64 with default blurring.

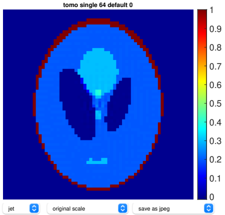


Figure: Single precision problem size 64 with default blurring.

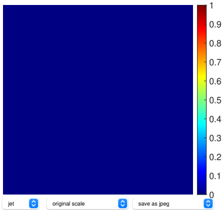


Figure: Half (fp16) precision problem size 64 with default blurring.

Experiment: Tomography (no noise)

- Why?

- NaNs!

ALGORITHM 7.4.1. CGLS. Let $x^{(0)}$ be an initial approximation, set

$$r^{(0)} = b - Ax^{(0)}, \quad p^{(0)} = s^{(0)} = A^T r^{(0)}, \quad \gamma_0 = \|s^{(0)}\|_2^2,$$

and for $k = 0, 1, 2, \ldots$ while $\gamma_k > tol$ compute

$$\begin{aligned}
q^{(k)} &= Ap^{(k)}, \\
\alpha_k &= \gamma_k / \|q^{(k)}\|_2^2, \\
x^{(k+1)} &= x^{(k)} + \alpha_k p^{(k)}, \\
r^{(k+1)} &= r^{(k)} - \alpha_k q^{(k)}, \\
s^{(k+1)} &= A^T r^{(k+1)}, \\
\gamma_{k+1} &= \|s^{(k+1)}\|_2^2, \\
\beta_k &= \gamma_{k+1} / \gamma_k, \\
p^{(k+1)} &= s^{(k+1)} + \beta_k p^{(k)}.
\end{aligned}$$

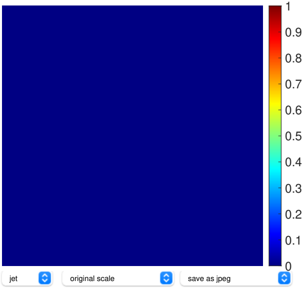
[3]

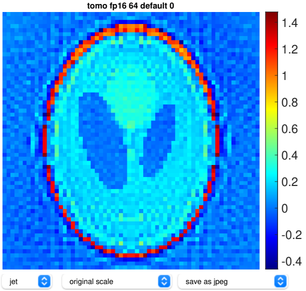
γ becomes Inf, there is an overflow.

[3] Björck, *Numerical methods for least squares problems*

Experiment: Tomography (no noise)

Our solution: A $\rightarrow A/100, b \rightarrow b/100$





Experiment: Tomography

- Error norm:

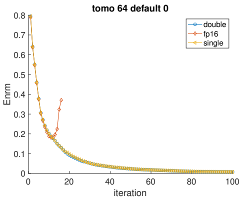


Figure: Error norm of size 64 problem with 0 noise at different precision level.

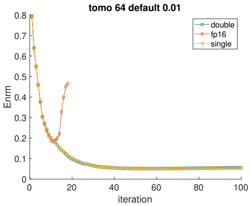


Figure: Error norm of size 64 problem with 1% noise at different precision level.

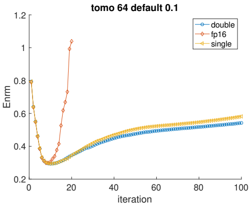


Figure: Error norm of size 64 problem with 10% noise at different precision level.

Experiment: Tomography (with noise)

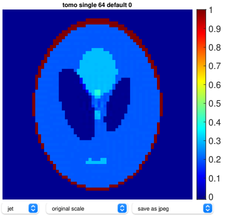


Figure: Single precision problem size 64 with zero noise.

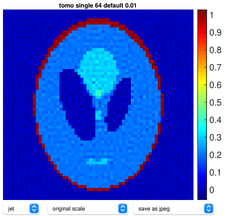


Figure: Single precision problem size 64 with 1% noise.

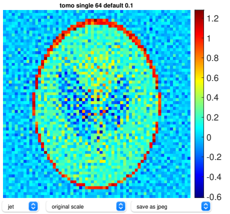


Figure: Single precision problem size 64 with 10% noise.

Experiment: Tomography (some interesting cases)

- γ becomes Inf in the original problem, the overflow results in NaNs from the first iteration
- After we divide both A and b by 10, $||q||_2^2$=Inf, $\alpha = x = 0$ in the first iteration. Later no underflow or overflow occurs, yet plot is always blur

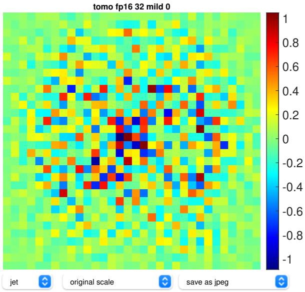


Figure: fp16 problem size 32 with zero noise

Experiment: Tomography (some interesting cases)

ALGORITHM 7.4.1. CGLS. Let $x^{(0)}$ be an initial approximation, set

$$r^{(0)} = b - Ax^{(0)}, \quad p^{(0)} = s^{(0)} = A^T r^{(0)}, \quad \gamma_0 = \|s^{(0)}\|_2^2,$$

and for $k = 0, 1, 2, \ldots$ while $\gamma_k > tol$ compute

$$\begin{aligned} q^{(k)} &= Ap^{(k)}, \\ \alpha_k &= \gamma_k / \|q^{(k)}\|_2^2, \\ x^{(k+1)} &= x^{(k)} + \alpha_k p^{(k)}, \\ r^{(k+1)} &= r^{(k)} - \alpha_k q^{(k)}, \\ s^{(k+1)} &= A^T r^{(k+1)}, \\ \gamma_{k+1} &= \|s^{(k+1)}\|_2^2, \\ \beta_k &= \gamma_{k+1}/\gamma_k, \\ p^{(k+1)} &= s^{(k+1)} + \beta_k p^{(k)}. \end{aligned}$$

[4]

$||q||_2^2$=Inf, $\alpha = x = 0$ in the first iteration.

[4] Björck, *Numerical methods for least squares problems*

Experiment: Tomography (some interesting cases)

- γ becomes Inf in the original problem, the overflow results in NaNs from the first iteration
- We set $A \to A/100$, and $b \to b/100$. This is the last iteration with all Inf and -Infs before NaN occurs.

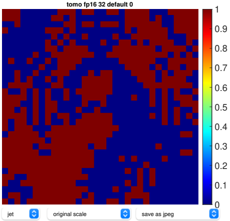


Figure: fp16 problem size 32 with default blur and zero noise, 14th iteration

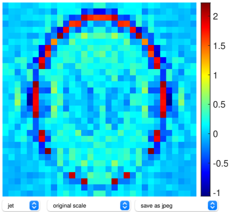


Figure: fp16 problem size 32 with default blur and zero noise, 13th iteration

Where We Will Go Next...

- Run experiments of larger sizes
- Implement other iterative methods that avoid inner products to eliminate NaNs

Bibliography

- Björck, Åke. *Numerical methods for least squares problems*. SIAM, 1996.
- *Using tensor cores for mixed-precision scientific computing*. Oct-2021. URL: https://developer.nvidia.com/blog/tensor-cores-mixed-precision-scientific-computing/.