Randomized Kaczmarz Algorithm

Math 881: Fall Class Project, 2020

Xiaowen(Lee) Li

Department of Mathematics
University of Kansas, USA
December 16, 2020

Contents

List of Figures

List of Tables

Abstract

Introduction

This report provides experiments about the performance of Randomized Kaczmarz's method that is introduced in the paper Strohmer and Vershynin (2009) : A randomized Kaczmarz algorithm with exponential convergence compares to Classical Randomized Kaczmarz's method and Simple Randomized Kaczmarz's method.

Experiments

This report contains following experiments: first, the rate of connvergence among all three versions of Kaczmarz's methods with input ramdomly draw from a nonuniform sampling distribution; Then repeating this experiment 100 times and analysising the least squares errors with fixed number of iterations as well as number of iterations with fixed degree of accuracy.

Conclusion

The Randomized Kaczmarz's method significantly outperforms the other Kaczmarz methods in all experiments which meets the result from the paper Strohmer and Vershynin (2009).

Introduction

Classical Kaczmarz's Method

Kaczmarz's method Kaczmarz (1937) is one of the most popular algorithm in solving the system of equations

$$Ax = b,$$

where A is a full rank $m \times n$ matrix with $m \geq n$, and $b \in \mathbb{C}^m$.

To denote the rows of A by $a_1^*, ..., a_m^*$ and let $b = (b_1, ..., b_m)^T$, The classical kaczmarz method picks the row of A in a cylic manner. The algorithm takes the form

$$x_{k+1} = x_k + \frac{b_i - < a_i, x_k >}{\|a_i\|_2^2} * a_i,$$

where $i = k \bmod m + 1$.

The theoretical estimates of the rate of converge of Kaczmarz's method are difficult to obtain for any matrix A with $m > 2$. All known estimates for certain examples are based on quantities of the matrix A which make the rate of convergence hard to compute and difficult to make comparision with convergence properties of other iterative methods.

Simple Randomized Kaczmarz's Method

To apply the idea from randomized algorithms to Classical Kaczmarz's method, Simple randomized Kaczmarz's Method use the rows of A in Kaczmarz's Method in random order, rather than in their given order. This algorithm takes the form

$$x_{k+1} = x_k + \frac{b_{s(i)} - < a_{s(i)}, x_k >}{\|a_{s(i)}\|_2^2} * a_{s(i)},$$

where $s(i)$ is choosen from the set $\{1, 2, ..., m\}$ at random.

No guarantees of its rate of convergence have been known.

Randomized Kaczmarz's Method

In the paper Strohmer and Vershynin (2009), authors proposed a new version of Randomized Kaczmarz's method, which picks the rows of matrix A randomly with probability proportional to the square of its Euclidean norm. This algorithm takes the form

$$x_{k+1} = x_k + \frac{b_{r(i)} - < a_{r(i)}, x_k >}{\|a_{r(i)}\|_2^2} * a_{r(i)},$$

where $r(i)$ is choosen from the set $\{1, 2, ..., m\}$ at random with probability proportional to $\|a_{r(i)}\|_2^2$.

And this randomized version of Kaczmarz's method can be proved with exponential expected rate of converegence.

Ideas behind choosing the probabilities according to the row-norms

- Choosing the probabilities according to the row-norms is related to the idea of preconditioning a matrix by row-scaling.
- From the viewpoint of preconditioning, it is clear that other methods of choosing a diagonal preconditioner will in general perform better.
- However, finding the optimal diagonal preconditioner for the system Ax = b can be very expensive while inverting the entire matrix A when A is large in size.
- Therefore, a cheaper, suboptimal alternative is needed. Scaling by the inverses of the squared row norms has been shown to be an efficient means to balance computational costs with optimality.

Theoretical Approach

Rate of convergence for Randomized Kazmarz Algorithm

Consider the linear system of equations $Ax = b$, where A is a full rank $m \times n$ matrix with $m \geq n$, and $b \in \mathbb{C}^m$.Let x be its solution. Then Randomized Kazcmarz Algorithm converges to x in expectation, with the average error

$$\mathbb{E}\|x_k - x\|_2^2 \leq (1 - \kappa(A)^{-2})^k \cdot \|x_0 - x\|_2^2$$

for all $k = 1, 2, ...$ Where $\kappa(A)$ is the scaled conditional number Demmel (1988) of matrix A that is defined as $\|A\|_F \cdot \|A^{-1}\|_2$.

Optimality for Randomized Kazmarz Algorithm

General Lower Estimate

Consider the same kind of linear system of equations as defined above, then there exists an initial approximation x_0 such that

$$\mathbb{E}\|x_k - x\|_2^2 \geq (1 - \frac{2k}{\kappa(A)^2}) \cdot \|x_0 - x\|_2^2$$

for all $k = 1, 2, ...$

The Upper Estimate is Attained

If $\kappa(A) = \sqrt{n}$, then

$$\mathbb{E}\|x_k - x\|_2^2 = (1 - \kappa(A)^{-2})^k \cdot \|x_0 - x\|_2^2$$

Discussion

From the above theoretical approaches, there are several advantages and disadvantages for Randomized Kaczmarz Method.

PROS

- The rate of convergence is expressible in terms of standard quantities in numerical analysis and does not depend on the number of equations in the linear system.
- In terms of accuracy, the average error has an upper bound.

CONS

- As stated above, the estimate cannot be improved beyond a constant factor.
- In terms of computational cost, the cost for computing all row norms of matrix A is $m \times n$ operations.

Numerical Experiments

Input

In order to show Randomized Kaczmarz Algorithm is efficient in general matrices. The experiment picked input randomly by a nonuniform sampling distribution. In this model, the problem has be formulated as follows: Define

$$f(t) = \sum_{l=-r}^{r} x_l e^{2\pi i l t},$$

where $x = \{x_l\}_{l=-r}^{r} \in \mathbb{C}^{2r+1}$.Gröchenig (1999)

Take nodes $\{t_k\}_{k=1}^{m}$ to be non-uniformly spaced by generating the sampling points t_j randomly between $[0, 1]$ and then sort them from smallest to largest. We arrive at the linear system of equations:

$$Ax = b,$$

where $A_{j,k} = \sqrt{w_j} e^{2\pi i l t_j}, b_j = \sqrt{w_j} f(t_j), w_j = \frac{t_{j+1} - t_{j-1}}{2}$,

with $j = 1, ...m; k = -r, ..., r; n = 2r + 1$.

In the experiment, it takes $r = 50, m = 700$, which makes A a 700×101 matrix.

Experiment 1

This experiment applies the Classical Kaczmarz's Method, the Simple Randomized Kaczmarz's Method, and the Randomized Kaczmarz's Method and plots the least squares error $\|x - x_k\|_2$ versus the number of projections (max iterations = 15000)s, cf. Figure 1.

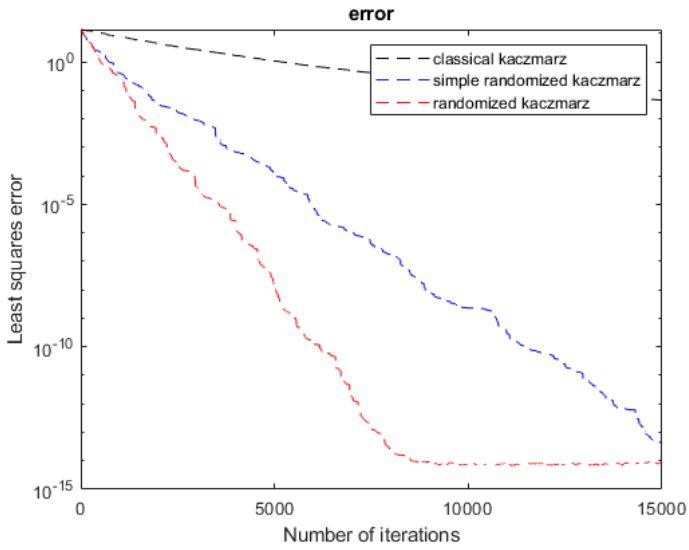

Figure 1: Comparison of rate of convergence for 3 methods

Observation

Figure 1 clearly shows that the Randomized Kaczmarz Algorithm converges much faster than the orther methods. It also generates smaller error than orther methods.

Experiment 2

This experiment applies all 3 methods with the described random input matrix A. Each method is terminated after reaching 5000 iterations. After repeating this experiment 100 times, the following histogram shows the least square error $\|x - x_{5000}\|_2$ for every method.

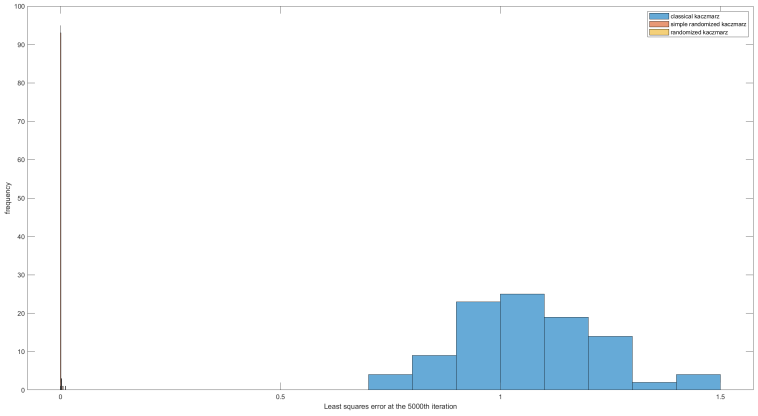

Figure 2: Error for 3 methods with fixed 5000 iterations in 100 runs

In figure 2, there is a big gap between errors generated by the Classical Kaczmarz's Method and the orther two randomized versions of Kaczmarz's method. I also came up with Figure 3 which shows the comparison within only two randomized versions of Kaczmarz's method.

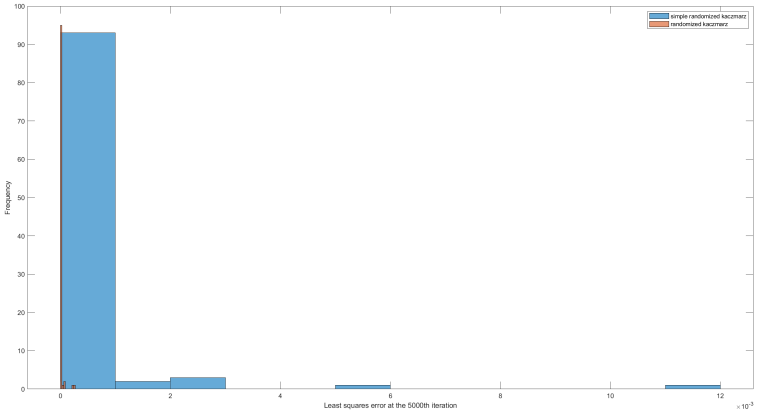

Figure 3: Error for 2 methods with fixed 5000 iterations in 100 runs

The statistical summary for above data is given below:

Table 1: Least squares error for 3 methods with fixed 5000 iterations

Classical	Simple Randomized	Randomized
Min. :0.702	Min. :0.00e+00	Min. :0.00e+00
1st Qu.:0.965	1st Qu.:2.00e-07	1st Qu.:1.00e-09
Median :1.059	Median :1.70e-06	Median :1.00e-09
Mean :1.073	Mean :2.85e-04	Mean :7.13e-06
3rd Qu.:1.169	3rd Qu.:1.78e-05	3rd Qu.:2.60e-08
Max. :1.498	Max. :1.12e-02	Max. :2.42e-04

Observation

From figure 2 and 3, we can tell that while the errors by Classical Kaczmarz's method is widely spreading between 0.5 to 1.5, even the errors by Simple Randomized Kaczmarz Algorithm is spreading bwtween 0 to 1.2e-2, the errors by the Randomized Kaczmarz Algorithm is squeezed close to 0.

Table 1 also shows how close the errors are for Randomized Kaczmarz Algorithm, with min to be 0 (this cannot be more accurate due to the limit of precision can be computed by MatLab), and the max to be as small as 2.42e-04 and a mean at 7.13e-06 which is 1e-02 more accurate than Simple Randomized Kaczmarz Algorithm and 1e-05 more accurate than Classical Kaczmarz Algorithm.

Experiment 3

This experiment applies all 3 methods with the described random input matrix A. Each method is terminated after reaching the required accuracy $\varepsilon = 10^{-4}$. After repeating this experiment 100 times, the following histogram shows the number of iterations until terminating for every method.

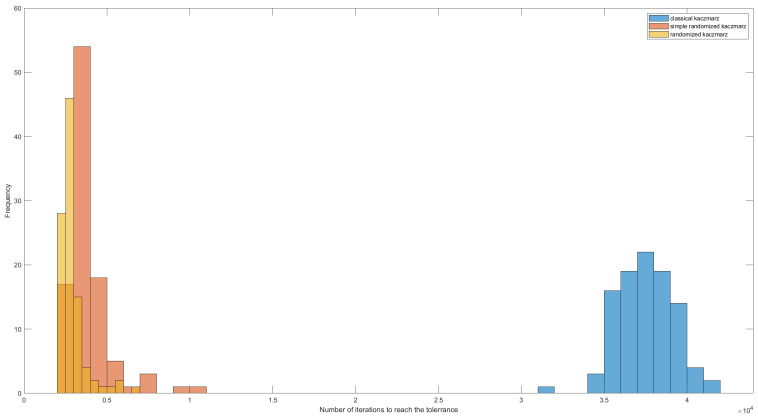

Figure 4: Number of iterations for 3 methods with fixed tolerrance in 100 runs

The statistical summary for above data is given below:

Table 2: Number of iterations for 3 methods with fixed tolerrance

Classical	Simple Randomized	Randomized
Min. :31010	Min. : 2647	Min. :2145
1st Qu.:36220	1st Qu.: 3162	1st Qu.:2486
Median :37538	Median : 3570	Median :2686
Mean :37509	Mean : 3926	Mean :2906
3rd Qu.:38791	3rd Qu.: 4182	3rd Qu.:3012
Max. :41391	Max. :10758	Max. :6973

Observation

Figure 4 shows that the number of iterations required for Randomized Kaczmarz Algorithm to reach the tolerrance is less than the Simple Randomized Kaczmarz Algorithm and much less than the Classical Kaczmarz Algorithm.

Table 2 also provide the same observation: Randomized Kaczmarz Algorithm has a mean value of iterations at 2906 while Simple Randomized Kaczmarz Algorithm's has it at 3926 and the Classical Kaczmarz Algorithm has it at 37509.

Conclusion

The Randomized Kaczmarz Algorithm significantly outperforms the other Kaczmarz methods both in the rate of convergence and the opration cost, demonstrating not only the power of choosing the projections at random but also the importance of choosing the projections according to their relevance.

Future Work

- We have shown that the Randomized Kaczmarz Algorithm is a very efficient way to work with general full rank matrices. But with some specifically determined matrices, there are known specific numerical methods to solve them, and it will be interesting to compare the performance of the Randomized Kaczmarz Algorithm with those methods with carefully structured inputs.
- All the theoretical analysis from the paper Strohmer and Vershynin (2009) and the experiments we did in this report assume the consistency of the system of equations - with some appropriate underrelaxation, it will be interesting to see if the Randomized Kaczmarz Algorithm can also do a great job in solving inconsistent systems.

Appendix: MatLab-code

Classical Kaczmarz Function

```
function [x,iter,error] = kaczmarz(A, b, x0,maxit,tol,exactx)
% classical kaczmarz
%
% Ax = b
% A - input matrix
% b - right vector
% x0 - initial x
%
% x - the approximated x
% iter - number of iterations before convergence (= maxit if maxit is given)
% error - the norm of difference in x and exact solution after every
% iteration

m = size(A,1);
n = size(A,2);

x = x0;
error = [];

if isempty(tol)
    iter = maxit;
    for i=1:maxit
       pickedi = mod(i,m)+1;
       row = A(pickedi, :);
       x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
       e = norm(x-exactx);
       error = [error,e];
    end
else
    iter = 1;
    e = 1;
    while e >= tol
       pickedi = mod(iter,m)+1;
       row = A(pickedi, :);
       x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
       e = norm(x-exactx);
       error = [error,e];
       iter = iter+1;
    end
end
end
```

Simple Randomized Kaczmarz Function

```
function [x,iter,error] = randkaczmarz(A, b, x0,maxit,tol,exactx)
% randomized kaczmarz by Algorithm 1
% Ax = b
% A - input matrix
% b - right vector
% x0 - initial x
%
% x - the approximated x
% iter - number of iterations before convergence (= maxit if maxit is given)
% error - the norm of difference in x and exact solution after every
% iteration
m = size(A,1);
n = size(A,2);

x = x0;
%iter = 0;
error = [];
e = 1;

normrow = [];
index = [];
%compute norm per row also store the corresponding index

  for i = 1:m
    normrow = [normrow,norm(A(i,:))];
    index = [index,i];
  end

  weight = normrow/sum(normrow);
if isempty(tol)
  iter = maxit;
  for i = 1:maxit
    %randsample to generate weighted random number from given vector
    pickedi = randsample(index,1,true,weight);

    row = A(pickedi, :);
    x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
    e = norm(x-exactx);
    error = [error,e];
    %iter = iter+1;
  end
else
    iter = 1;
    e = 1;
    while e >= tol
```

```
    %randsample to generate weighted random number from given vector
    pickedi = randsample(index,1,true,weight);

    row = A(pickedi, :);
    x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
    e = norm(x-exactx);
    error = [error,e];
    iter = iter+1;
    end
end
end
```

Randomized Kaczmarz Function

```
function [x,iter,error] = simplerandkaczmarz(A, b, x0,maxit,tol,exactx)
% simplerandomized kaczmarz mentioned by fig 1
% Ax = b
% A - input matrix
% b - right vector
% x0 - initial x
%
% x - the approximated x
% iter - number of iterations before convergence (= maxit if maxit is given)
% error - the norm of difference in x and exact solution after every
% iteration
m = size(A,1);
n = size(A,2);

x = x0;
%iter = 0;
error = [];
e = 1;

if isempty(tol)
  iter = maxit;
  for i = 1:maxit
    %randsample to generate weighted random number from given vector
    pickedi = randi(m);

    row = A(pickedi, :);
    x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
    e = norm(x-exactx);
    error = [error,e];
    %iter = iter+1;
  end
else
```

```
    iter = 1;
    e = 1;
    while e >= tol
        pickedi = randi(m);

    row = A(pickedi, :);
    x = x + ( b(pickedi) - (row * x) ) / (row * row') * row';
    e = norm(x-exactx);
    error = [error,e];
    iter = iter+1;
    end
end
end
```

Input Function

```
function [classical_iter_or_error,simple_iter_or_error,rand_iter_or_error] = Input(iter,tol)
%% set the dimention
r = 50;
m = 700;
n = 2*r+1;

%% generate nodes
%tj are drawing randomly from a uniform distribution in [0,1]
t = rand(m,1);

%then ordering by magnitude (since they are all positive, just sort)
t = sort(t);
% just to assign the size
w = zeros(m,1);
x = zeros(n,1);
%% generate x
realx = randn(n,1);
imgx = randn(n,1);
for l = 1:n
   x(l) = realx(l)+1i*imgx(l);
end
%% generate A and b
A = zeros(m,n);
for j = 1:m
    % dealing with special cases when reach the endpoints(nodes)
    if j == 1
        w(j) = (t(2)-t(end)-1)/2;
    elseif j == m
           w(j) = (t(1)+1-t(m-1))/2;
        else
```

```
           w(j) = (t(j+1)-t(j-1))/2;
    end
    for k = -r:r
       A(j,k+r+1) =sqrt(w(j))*exp(2*pi*1i*k*t(j));
    end
end
b = A*x;
x0 = zeros(n,1);

[x1,iter1,error1] = kaczmarz(A,b,x0,iter,tol,x);
[x2,iter2,error2] = simplerandkaczmarz(A,b,x0,iter,tol,x);
[x3,iter3,error3] = randkaczmarz(A,b,x0,iter,tol,x);
if isempty(tol)
    classical_iter_or_error = error1;
    simple_iter_or_error = error2;
    rand_iter_or_error = error3;
else
    classical_iter_or_error = iter1;
    simple_iter_or_error = iter2;
    rand_iter_or_error = iter3;
end
```

Driver

```
%% experiment 1 (presentation) comparing rate of convergence among 3 versions with
%maxit = 15000
%(same experiment as from the original paper)
[error1,error2,error3] = Input(15000,[]);

figure (1)
semilogy(1:maxit,error1,'k--');
title('comparing rate of convergence among 3 versions with maxit = 15000')
ylabel('Least squares error')
xlabel('Number of iterations')
hold on
semilogy(1:maxit,error2,'b--');
semilogy(1:maxit,error3,'r--');
hold off
legend('classical kaczmarz','simple randomized kaczmarz','randomized kaczmarz')
%% experiment 2 (report) Running 100 times with random input and maxit = 5000
%- statistical analysis
error_c = [];
error_s = [];
error_r = [];
for n = 1:100
    [error1,error2,error3] = Input(5000,[]);
```

```
    error_c = [error_c,error1(5000)];
    error_s = [error_s,error2(5000)];
    error_r = [error_r,error3(5000)];
end
% histogram
figure (2)
title('errors')
h3 = histogram(error_c);
hold on
h4 = histogram(error_s);
h5 = histogram(error_r);
hold off
legend('classical kaczmarz','simple randomized kaczmarz','randomized kaczmarz')


figure (3)
title('errors')
h1 = histogram(error_s);
hold on
h2 = histogram(error_r);
hold off
legend('simple randomized kaczmarz','randomized kaczmarz')

%% experiment 3 (report) Running 100 times with random input and tol = 10^(-4)
%- statistical analysis
iter_c = [];
iter_s = [];
iter_r = [];
for n = 1:100
    [iter1,iter2,iter3] = Input([],0.0001);
    iter_c = [iter_c,iter1];
    iter_s = [iter_s,iter2];
    iter_r = [iter_r,iter3];
end
% histogram
figure (4)
title('iterations')
h3 = histogram(iter_c);
hold on
h4 = histogram(iter_s);
h5 = histogram(iter_r);
hold off
legend('classical kaczmarz','simple randomized kaczmarz','randomized kaczmarz')
figure (5)
title('iterations')
h1 = histogram(iter_s);
hold on
```

```
h2 = histogram(iter_r);
hold off
legend('simple randomized kaczmarz','randomized kaczmarz')
```

Appendix: R-code

```
library(knitr)
library(formatR)
library(stargazer)
library(xtable)
library(png)
library(R.matlab)
knitr::opts_chunk$set(echo = TRUE)
options(digits = 5, width = 60, xtable.comment = FALSE)
opts_chunk$set(tidy.opts = list(width.cutoff=60), tidy=TRUE)
out_type <- knitr::opts_knit$get("rmarkdown.pandoc.to")
#Figure 1
img1_path <- "images/1.png"
include_graphics(img1_path)
#Figure 2
img2_path <- "images/2.png"
include_graphics(img2_path)
#Figure 3
img3_path <- "images/3.png"
include_graphics(img3_path)
#Figure 4
img4_path <- "images/4.png"
include_graphics(img4_path)
#Table 1
errorc <- readMat("data/errorc.mat")
errors <- readMat("data/errors.mat")
errorr<- readMat("data/errorr.mat")
c_name <- "Classical"
s_name <- "Simple Randomized"
r_name <- "Randomized"

df <- data.frame(t(errorc$error.c),t(errors$error.s),t(errorr$error.r))
names(df) <- c(c_name,s_name,r_name)
kable(summary(df),caption = "Least squares error for 3 methods")
#Table 2
iterc <- readMat("data/iterc.mat")
iters <- readMat("data/iters.mat")
iterr <- readMat("data/iterr.mat")
c_name <- "Classical"
s_name <- "Simple Randomized"
r_name <- "Randomized"

dff <- data.frame(t(iterc$iter.c),t(iters$iter.s),t(iterr$iter.r))
names(dff) <- c(c_name,s_name,r_name)

kable(summary(dff),caption = "Number of iterations for 3 methods with fixed tolerrance")
```

Bibliography

Demmel, J. W. (1988), "The probability that a numerical analysis problem is difficult," *Mathematics of Computation*, 50, 449–480.

Gröchenig, K. (1999), "Irregular sampling, toeplitz matrices, and the approximation of entire functions of exponential type," *Mathematics of Computation*, 68, 749–765.

Kaczmarz, S. (1937), "Angenaherte auflosung von systemen linearer gleichungen: Bulletin international de l'académie polonaise des sciences et des lettres," .

Strohmer, T. and Vershynin, R. (2009), "A randomized kaczmarz algorithm with exponential convergence," *Journal of Fourier Analysis and Applications*, 15, 262.