Code of a multidimensional fractional quasi-Newton method with an order of convergence at least quadratic using recursive programming

A. Torres-Hernandez ∗,a

aDepartment of Physics, Faculty of Science - UNAM, Mexico

Abstract

The following paper presents a way to define and classify a family of fractional iterative methods through a group of fractional matrix operators, as well as a code written in recursive programming to implement a variant of the fractional quasi-Newton method, which through minor modifications, can be implemented in any fractional fixed-point method that allows solving nonlinear algebraic equation systems.

Keywords: Fractional Operators; Group Theory; Fractional Iterative Methods; Recursive Programming.

1. Fractional Quasi-Newton Method Accelerated

To begin this section, it is necessary to mention that due to the large number of fractional operators that may exist [1–13], some sets must be defined to fully characterize the fractional quasi-Newton method accelerated [14,15]. It is worth mentioning that characterizing elements of fractional calculus through sets is the main idea behind of the methodology known as fractional calculus of sets [16]. So, considering a scalar function $h : \mathbb{R}^m \to \mathbb{R}$ and the canonical basis of $\mathbb{R}^m$ denoted by $\{\hat{e}_k\}_{k\geq 1}$, it is possible to define the following fractional operator of order $\alpha$ using Einstein notation

$$o^\alpha_x h(x) := \hat{e}_k o^\alpha_k h(x).$$

Therefore, denoting by $\partial^n h$ the partial derivative of order $n$ applied with respect to the $k$-th component of the vector $x$, using the previous operator it is possible to define the following set of fractional operators

$$O^n_{x,a} (h) := \{ o^\alpha_x : \exists o^\alpha_k h(x) \text{ and } \lim_{a \to n} o^\alpha_k h(x) = \partial^n_k h(x) \forall k \geq 1 \},$$

whose complement may be defined as follows

$$O^{n,c}_{x,a} (h) := \{ o^\alpha_x : \exists o^\alpha_k h(x) \forall k \geq 1 \text{ and } \lim_{a \to n} o^\alpha_k h(x) \neq \partial^n_k h(x) \text{ in at least one value } k \geq 1 \},$$

as a consequence, it is possible to define the following set

$$O^{n,u}_{x,a} (h) := O^n_{x,a} (h) \cup O^{n,c}_{x,a} (h).$$

On the other hand, considering a function $h : \Omega \subset \mathbb{R}^m \to \mathbb{R}^m$, it is possible to define the following set

$$m_1 O^{n,u}_{x,a} (h) := \{ o^\alpha_x : o^\alpha_x \in O^{n,u}_{x,a} ([h]_k) \forall k \leq m \},$$

∗E-mail: anthony.torres@ciencias.unam.mx; ORCID: 0000-0001-6496-9505

†Método quasi-Newton fraccional acelerado.

DOI : 10.5121/mathsj.2022.9103
where \([h]_k : \Omega \subset \mathbb{R}^m \rightarrow \mathbb{R}\) denotes the \(k\)-th component of the function \(h\). So, it is possible to define the following set of fractional operators

\[
m \text{MO}^{\alpha\theta}_{x,a}(h) := \bigcap_{k \in \mathbb{Z}} mQ^{k,a}_{x,a}(h),
\]

which under the classical Hadamard product it is fulfilled that

\[
o^{\alpha\theta}_x \circ h(x) := h(x) \forall o^{\alpha\theta}_x \in m \text{MO}^{\alpha\theta}_{x,a}(h).
\]

Then, considering that for each operator \(o^{\alpha\theta}_x\) it is possible to define the following fractional matrix operator

\[
A(a(o^{\alpha\theta}_x)) = \left([A(a(o^{\alpha\theta}_x)))_{jk} = (o^{\alpha\theta}_k\right).
\]

it is possible to define for each operator \(o^{\alpha\theta}_x \in m \text{MO}^{\alpha\theta}_{x,a}(h)\) the following matrix

\[
A_{h,a} := A(a(o^{\alpha\theta}_x)) \circ A^T_a(h),
\]

where \(A_a(h) = ([A(a(h))]_{jk} = ([h]_k)\). On the other hand, considering that when using the classical Hadamard product in general \(o^{\alpha\theta}_x \circ o^{\alpha\theta}_x \neq o^{(\alpha+\theta)}_x\). It is possible to define the following modified Hadamard product [16]:

\[
o^{p\alpha}_{i,x} \circ o^{q\alpha}_{j,x} := \begin{cases} o^{p\alpha}_{i,x} \circ o^{q\alpha}_{j,x}, & \text{if } i \neq j \text{ (Hadamard product of type horizontal)} \\ o^{(p+q)\alpha}_{i,x}, & \text{if } i = j \text{ (Hadamard product of type vertical)} \end{cases},
\]

with which it is possible to obtain the following theorem:

**Theorem 1.** Let \(o^{\alpha\theta}_x\) be a fractional operator such that \(o^{\alpha\theta}_x \in m \text{MO}^{\alpha\theta}_{x,a}(h)\). So, considering the modified Hadamard product given by (10), it is possible to define the following set of fractional matrix operators

\[
m G(A(a(o^{\alpha\theta}_x))) := \left[A^{\alpha\theta}_a = A(a(o^{\alpha\theta}_x)) : r \in \mathbb{Z} \text{ and } A^{\alpha\theta}_a = ([A^{\alpha\theta}_a])_{jk} = (o^{\alpha\theta}_k) \right]
\]

which corresponds to the Abelian group generated by the operator \(A_a(o^{\alpha\theta}_x)\).

**Proof.** It should be noted that due to the way the set (11) is defined, just the Hadamard product of type vertical is applied among its elements. So, \(\forall A^{\alpha\theta}_a, A^{\alpha\theta}_a \in m G(A_a(o^{\alpha\theta}_x))\) it is fulfilled that

\[
A^{\alpha\theta}_a \circ A^{\alpha\theta}_a = ([A^{\alpha\theta}_a(1)])_{jk} = (o^{(p+q)\alpha}_k) = A^{\alpha\theta}_a(1),
\]

with which it is possible to prove that the set (11) fulfills the following properties, which correspond to the properties of an Abelian group:

\[
\begin{align*}
\forall A^{\alpha\theta}_a, A^{\alpha\theta}_a, A^{\alpha\theta}_a & \in m G(A_a(o^{\alpha\theta}_x)) \text{ it is fulfilled that } A^{\alpha\theta}_a \circ A^{\alpha\theta}_a \circ A^{\alpha\theta}_a = A^{\alpha\theta}_a \circ A^{\alpha\theta}_a \circ A^{\alpha\theta}_a, \\
\exists A^{\alpha\theta}_a & \in m G(A_a(o^{\alpha\theta}_x)) \text{ such that } \forall A^{\alpha\theta}_a \in m G(A_a(o^{\alpha\theta}_x)) \text{ it is fulfilled that } A^{\alpha\theta}_a \circ A^{\alpha\theta}_a = A^{\alpha\theta}_a, \\
\forall A^{\alpha\theta}_a & \in m G(A_a(o^{\alpha\theta}_x)) \exists A^{\alpha\theta}_a \in m G(A_a(o^{\alpha\theta}_x)) \text{ such that } A^{\alpha\theta}_a \circ A^{\alpha\theta}_a = A^{\alpha\theta}_a, \\
\forall A^{\alpha\theta}_a, A^{\alpha\theta}_a & \in m G(A_a(o^{\alpha\theta}_x)) \text{ it is fulfilled that } A^{\alpha\theta}_a \circ A^{\alpha\theta}_a = A^{\alpha\theta}_a.
\end{align*}
\]
From the previous theorem, it is possible to define the following group of fractional matrix operators [16]:

$$mG_{FIM}(\alpha) := \bigcup_{\alpha^q \in \mathbb{M}G_{FIM}(\alpha)} A_{i,a}(\alpha^q),$$

(14)

where $\forall A_{i,a}, A_{j,a} \in mG_{FIM}(\alpha)$, with $i \neq j$, the following property is defined

$$A_{i,a}^{\alpha} \circ A_{j,a}^{\alpha} = A_{k,a}^{\alpha} := A_{k,a}(\alpha_{i,x}^{\alpha,\beta} \circ \alpha_{j,x}^{\alpha,\beta}), \quad p, q \in \mathbb{Z} \setminus \{0\},$$

as a consequence, it is fulfilled that

$$\forall A_{k,a}^{\alpha} \in mG_{FIM}(\alpha) \text{ such that } A_{k,a}(\alpha_{i,x}^{\alpha,\beta} \circ \alpha_{j,x}^{\alpha,\beta}) \exists A_{k,a}^{\alpha r} = A_{k,a}^{\alpha(r-1)} \circ A_{k,a}^{\alpha 1} = A_{k,a}(\alpha_{i,x}^{\alpha,\beta} \circ \alpha_{j,x}^{\alpha,\beta}).$$

(16)

Therefore, if $\Phi_{FIM}$ denotes the iteration function of some fractional iterative method [16], it is possible to obtain the following result:

Let $\alpha_0 \in \mathbb{R} \setminus \mathbb{Z} \Rightarrow A_{\alpha_0}^{\alpha} \in mG_{FIM}(\alpha) \exists \Phi_{FIM} = \Phi_{FIM}(A_{\alpha_0}) : \forall A_{\alpha_0} \exists \{\Phi_{FIM}(A_{\alpha}) : \alpha \in \mathbb{R} \setminus \mathbb{Z}\}.$

(17)

So, from the previous result, it is possible to define different sets that allow characterizing different fractional iterative methods. For example, the fractional Newton-Raphson method may be characterized through the following set [16,17]:

$$mG_{FN}(\alpha) := mG_{FIM}(\alpha) \cap \{\alpha_{i,x}^{\alpha,\beta} \exists A_{h,a}^{-1} = A_{a}^{\alpha} \circ A_{a}^{\alpha T} (h)\},$$

(18)

while the fractional pseudo-Newton method may be characterized through the following set [18,19]:

$$mG_{FPN}(\alpha) := mG_{FIM}(\alpha) \cap \{\alpha_{i,x}^{\alpha,\beta} \exists \alpha_{i,x}^{\alpha,\beta} c \neq 0 \forall c \in \mathbb{R} \setminus \{0\} \text{ and } \forall k \geq 1\},$$

(19)

as a consequence, the fractional quasi-Newton method may be characterized through the following set of fractional matrix operators [14,20]:

$$mG_{FQN}(\alpha) := mG_{FN}(\alpha) \cap mG_{FPN}(\alpha).$$

(20)

Before continuing it is necessary to define the following corollary:

**Corollary 1.** Let $f : \Omega \subset \mathbb{R}^m \rightarrow \mathbb{R}^m$ be a function with a point $\xi \in \Omega$ such that $\|f(\xi)\| = 0$, and let $h : \Omega \subset \mathbb{R}^m \rightarrow \mathbb{R}^m$ be a function such that $h^{1}(x) = f^{1}(x) \forall x \in B(\xi; \delta).$ So, $\forall \alpha^{q} \in mG_{FIM}(\alpha)$ such that $A_{\alpha}^{\alpha} (\alpha_{i,x}^{\alpha,\beta}) \in mG_{FN}(\alpha)$, there exists $A_{h,a}^{-1} = A_{a}^{\alpha} (\alpha_{i,x}^{\alpha,\beta}) \circ A_{a}^{\alpha T} (h)$ such that it fulfills the following condition

$$\lim_{\alpha \rightarrow 1} A_{h,a}(x) = \left(f^{1}(x)\right)^{-1} \forall x \in B(\xi; \delta).$$

(21)

Then, defining the following function

$$a_{f}([k], x) := \begin{cases} \alpha, & \text{if } \|[k]_x\| \neq 0 \text{ and } \|f(x)\| > \delta_0 \\ 1, & \text{if } \|[k]_x\| = 0 \text{ or } \|f(x)\| \leq \delta_0 \end{cases},$$

(22)

the fractional quasi-Newton method accelerated may be defined and classified through the following set of matrices [15]:

$$\left\{ A_{h,a}^{\alpha} = A_{h,a}^{\alpha T} (A_{\alpha}^{\alpha}) : A_{\alpha}^{\alpha} \in mG_{FQN}(\alpha) \text{ and } \lim_{\alpha \rightarrow 1} A_{h,a}(x) = \left(f^{1}(x)\right)^{-1} \text{ with } A_{h,a}(x) = \left([A_{h,a}\alpha]_{jk}(x)\right) \right\}. \quad (23)$$

To end this section, it is worth mentioning that the fractional quasi-Newton method accelerated has been used in the study for the construction of hybrid solar receivers [15], and that in recent years there has been a growing interest in fractional operators and their properties for solving nonlinear algebraic equation systems [17,21–28].
2. Programming Code of Fractional Quasi-Newton Method Accelerated

The following code was implemented in Python 3 and requires the following packages:

```python
import math as mt
import numpy as np
from numpy import linalg as la
```

For simplicity, a two-dimensional vector function is used to implement the code, that is, \( f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^2 \), which may be denoted as follows:

\[
f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix},
\]  

(24)

where \([f]_i : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R} \ \forall i \in \{1, 2\} \). Then considering a function \( \Phi : (\mathbb{R}\setminus \mathbb{Z}) \times \mathbb{C}^n \rightarrow \mathbb{C}^n \), a variant of the fractional quasi-Newton method may be denoted as follows [15, 16]

\[
x_{i+1} := \Phi(\alpha, x_i) = x_i - A_{h_f, \alpha}(x_i)f(x_i), \quad i = 0, 1, 2 \cdots ,
\]

(25)

where \( A_{h_f, \alpha}(x_i) \) is a matrix evaluated in the value \( x_i \), which is given by the following expression

\[
A_{h_f, \alpha}(x_i) = \left( [A_{h_f, \alpha}]_{jk}(x_i) \right) := \left( \frac{\partial}{\partial x_j} \left( f_i(x) \right) \right)_{x_i}^{-1},
\]

(26)

with \( h_f(x) := f(x_i) + f^{(1)}(x_i)(x-x_i) \). It is worth mentioning that one of the main advantages of fractional iterative methods is that the initial condition \( x_0 \) can remain fixed, with which it is enough to vary the order \( \alpha \) of the fractional operators involved until generating a sequence convergent \( \{x_i\}_{i \geq 1} \) to the value \( \xi \in \Omega \). Since the order \( \alpha \) of the fractional operators is varied, different values of \( \alpha \) can generate different convergent sequences to the same value \( \xi \) but with a different number of iterations. So, it is possible to define the following set

\[
\text{Conv}_\alpha(\xi) := \left\{ \Phi : \lim_{x \rightarrow \xi} \Phi(\alpha, x) = \xi \in B(\xi, \delta) \right\},
\]

(27)

which may be interpreted as the set of fractional fixed-point methods that define a convergent sequence \( \{x_i\}_{i \geq 1} \) to some value \( \xi \in B(\xi, \delta) \). So, denoting by \( \text{card}(\cdot) \) the cardinality of a set, under certain conditions it is possible to prove the following result (see reference [16], proof of Theorem 2):

\[
\text{card}(\text{Conv}_\alpha(\xi)) = \text{card}(\mathbb{R}),
\]

(28)

from which it follows that the set (27) is generated by an uncountable family of fractional fixed-point methods. Before continuing, it is necessary to define the following corollary [16]:

**Corollary 2.** Let \( \Phi : (\mathbb{R}\setminus \mathbb{Z}) \times \mathbb{C}^n \rightarrow \mathbb{C}^n \) be an iteration function such that \( \Phi \in \text{Conv}_\alpha(\xi) \). So, if \( \Phi \) has an order of convergence of order (at least) \( p \) in \( B(\xi; 1/2) \), for some \( m \in \mathbb{N} \), there exists a sequence \( \{P_i\}_{i \geq m} \in B(p; \delta_\alpha) \) given by the following values

\[
P_i = \frac{\log(\|x_{i-1} - x_i\|)}{\log(\|x_{i-2} - x_{i-1}\|)},
\]

(29)

such that it fulfills the following condition:

\[
\lim_{i \rightarrow \infty} P_i \rightarrow p,
\]

and therefore, there exists at least one value \( k \geq m \) such that

\[
P_k \in B(p; \epsilon).
\]

(30)
The previous corollary allows estimating numerically the order of convergence of an iteration function $\Phi$ that generates at least one convergent sequence $\{x_i\}_{i \geq 1}$. On the other hand, the following corollary allows characterizing the order of convergence of an iteration function $\Phi$ through its Jacobian matrix $\Phi^{(1)}$ [16,28]:

**Corollary 3.** Let $\Phi : (\mathbb{R} \setminus \mathbb{Z}) \times \mathbb{C}^n \rightarrow \mathbb{C}^n$ be an iteration such that $\Phi \in \text{Conv}_\delta(\xi)$. So, if $\Phi$ has an order of convergence of order (at least) $p$ in $B(\xi; \delta)$, it is fulfilled that:

$$p := \begin{cases} 1, & \text{if } \lim_{x \to \xi} \|\Phi^{(1)}(a,x)\| = 0 \\ 2, & \text{if } \lim_{x \to \xi} \|\Phi^{(1)}(a,x)\| = 0 \end{cases}$$  \hspace{1cm} (31)

Before continuing, it is necessary to mention that what is shown below is an extremely simplified way of how a fractional iterative method should be implemented, a more detailed description, as well as some applications, may be found in the references [14–18, 28–30]. Considering the following notation:

$$\text{ErrDom} := \left\{ \|x_i - x_{i-1}\|_2 \right\}_{i \geq 1}, \quad \text{ErrIm} := \left\{ \|f(x_i)\|_2 \right\}_{i \geq 1}, \quad X := \left\{ x_i \right\}_{i \geq 1},$$  \hspace{1cm} (32)

it is possible to implement a particular case of the multidimensional fractional quasi-Newton method accelerated through recursive programming using the following functions:

```python
def Dfrac(a, µ, x):
    s = µ - α
    if µ > -1:
        return (mt.gamma(µ+1)/mt.gamma(s+1)) * pow(complex(x), s) if mt.ceil(s) - s > 0 or s > -1 else 0

def αf(α, xk, normf):
    δ0 = 3
    return α if abs(xk) > 0 and normf > δ0 else 1

def FractionalQuasiNewton(ErrDom, ErrIm, X, α, x0):
    Tol = pow(10, -5)
    Lim = pow(10, 2)
    InvA = InvAhf(α, x0)
    if abs(la.det(InvA)) > 0:
        x1 = x0 - np.matmul(la.inv(InvA), f(x0))
        ED = la.norm(x1 - x0)
        if ED > 0:
            EI = la.norm(f(x1))
            ErrDom.append(ED)
            ErrIm.append(EI)
            X.append(x1)
            N = len(X)
            if max(ED, EI) > Tol and N < Lim:
                ErrDom, ErrIm, X = FractionalQuasiNewton(ErrDom, ErrIm, X, α, x1)
    return ErrDom, ErrIm, X
```

To implement the above functions, it is necessary to follow the steps shown below:

i) A function must be programmed together with its Jacobian matrix.

```python
def f(x):
    y = np.zeros((2, 1)).astype(complex)
    y[0] = np.sin(x[0]) * pow(x[0], 2) + np.cos(x[1]) * pow(x[1], 3) - 5
    y[1] = np.cos(x[0]) * pow(x[0], 3) - np.sin(x[1]) * pow(x[1], 2) - 7
    return y

def Df(x):
    y = np.zeros((2, 2)).astype(complex)
    y[0][0] = 2 * np.sin(x[0]) * x[0] + np.cos(x[0]) * pow(x[0], 2)
    y[0][1] = 3 * np.cos(x[1]) * pow(x[1], 2) - np.sin(x[1]) * pow(x[1], 3)
    y[1][0] = 3 * np.cos(x[0]) * pow(x[0], 2) - np.sin(x[0]) * pow(x[0], 3)
    y[1][1] = -2 * np.sin(x[1]) * x[1] - np.cos(x[1]) * pow(x[1], 2)
    return y
```
ii) The matrix $A_{h_f,a_f}^{-1}$ must be programmed.

```python
def InvAhf(a, x):
    f0 = f(x)
    Df0 = Df(x)
    normf = la.norm(f0)
    h1 = f0[0]
    h1x = Df0[0][0]
    h1y = Df0[0][1]
    h21 = f0[1]
    h2x = Df0[1][0]
    h2y = Df0[1][1]
    a1 = a[f(a, x[0]), normf]
    a2 = a[f(a, x[1]), normf]
    y = np.zeros((2, 2)).astype(complex)
    y[0][0] = (h1 - h1x * x[0]) * Df0[0][0] + h1y * Df0[0][1]
    y[0][1] = (h21 - h2x * x[0]) * Df0[1][0] + h2y * Df0[1][1]
    return y
```

iii) Three empty vectors, a fractional order $\alpha$, and an initial condition $x_0$ must be defined before implementing the function FractionalQuasiNewton.

```python
ErrDom = []
ErrIm = []
X = []
alpha = -1.598394
x0 = 2.25 * np.ones((2, 1))
ErrDom, ErrIm, X = FractionalQuasiNewton(ErrDom, ErrIm, X, a, x0)
```

When implementing the previous steps, if the fractional order $\alpha$ and initial condition $x_0$ are adequate to approach a zero of the function $f$, results analogous to the following are obtained:

<table>
<thead>
<tr>
<th>i</th>
<th>$x_{i1}$</th>
<th>$x_{i2}$</th>
<th>$|x_{i1}-x_{i1}|$</th>
<th>$|x_{i2}-x_{i2}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.735585327165831</td>
<td>-1.070268335056107</td>
<td>7.7345060713871</td>
<td>18.83715358624463</td>
</tr>
<tr>
<td>2</td>
<td>-4.73571464234244 -0.000520175227302893</td>
<td>-1.4523845601521614 -1.1965490706379078</td>
<td>1.2560746005194463</td>
<td>15.55066639012087</td>
</tr>
<tr>
<td>3</td>
<td>-4.7371740194018076 -0.0028188821384910173</td>
<td>-1.3067374770351186 -1.1704706967673722</td>
<td>0.1479856484317521</td>
<td>18.2335900945263</td>
</tr>
<tr>
<td>4</td>
<td>-4.7380708853423333 -0.00490292129265445</td>
<td>-1.05437668532889 -1.1528378083779465</td>
<td>0.252886370176711</td>
<td>21.80092139297337</td>
</tr>
<tr>
<td>5</td>
<td>-4.738147107048297 -0.00657969269076015</td>
<td>-0.538288869737178 -1.218835356081686</td>
<td>0.520934934794708</td>
<td>25.019899629988</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>-4.72975025224614 -0.004361695131642888</td>
<td>0.597441863428369 -1.7216488909240522</td>
<td>0.0063439842485266</td>
<td>3.004542013733636</td>
</tr>
<tr>
<td>13</td>
<td>-4.7286434095025385 -0.0039712413830369586</td>
<td>0.5990564198651102 -1.724723647139149</td>
<td>0.0033606622313366525</td>
<td>2.9995701108095517</td>
</tr>
<tr>
<td>14</td>
<td>-4.730316596597024 +0.0024479642825272506</td>
<td>0.5937988516457116 -1.7278144278472323</td>
<td>0.028695509313230096</td>
<td>0.05460293489272963</td>
</tr>
<tr>
<td>15</td>
<td>-4.7308693163010615 +0.004410915392472121</td>
<td>0.593900826713869 -1.7280910966503522</td>
<td>0.000634169330564483</td>
<td>2.0574246244311213e-05</td>
</tr>
<tr>
<td>16</td>
<td>-4.7308693163010615 +0.004410915392472121</td>
<td>0.593900826713869 -1.7280910966503522</td>
<td>0.000634169330564483</td>
<td>2.0574246244311213e-05</td>
</tr>
</tbody>
</table>

Table 1: Results obtained using the fractional quasi-Newton method accelerated [15].

Therefore, from the Corollary 2, the following result is obtained:

$$P_{16} = \frac{\log(\|x_{16} - x_{15}\|)}{\log(\|x_{15} - x_{14}\|)} = 2.0361 \in B(p; \delta_K),$$

which is consistent with the Corollary 3, since if $\Phi_{FQN} \in \text{Conv}_\delta(\xi)$, in general $\Phi_{FQN}(A_{h_f,a_f})$ fulfills the following condition (see reference [28], proof of Proposition 1):

$$\lim_{x \to \xi} \|\Phi_{FQN}^{(1)}(1,x)\| = 0,$$

from which it is concluded that the fractional quasi-Newton method accelerated has an order of convergence (at least) quadratic in $B(\xi; \delta)$. 


References


