QUASI-MINIMAL RESIDUAL ALGORITHM

Sarina Heshmati

sarinaheshmatii@gmail.com

INTRODUCTION

INTRODUCTION

- The new world needs new tools in order to solve its problems.
- The tools need to efficient, precise and simple.
- QMR is one of the answers.

BRIEF EXPLANATION

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

WHAT IS QMR?

WHAT IS QMR?

Quasi

WHAT IS QMR?

Minimal

WHAT IS QMR?

Residual

WHAT IS QMR?

WHAT IS QMR?

- Iterative method as an improvement over the MINRES algorithm
- Minimizing the residual by projecting it onto a Krylov subspace
- The Difference with CG algorithm
- Can be improved by using the appropriate preconditioners

THE ALGORITHM

THE STEPS FOR SOLVING: $Ax = b$

1.

3.

a. Krylov subspace vectors
b. Step sizes
c. Update the solution
d. Update the residual
e. Left & right preconditioned residuals
f. Update the search directions

a.
$Ay_k, A^T z_k$

b.
$$\alpha_k = \frac{(r_k, z_k)}{(y_k, Ay_k)}$$
$$\beta_k = \frac{(r_k, A^T z_k)}{(y_k, A^T z_k)}$$

c.
$$x_{k+1} = x_k + \alpha_k y_k$$

d.
$$r_{k+1} = r_k - \alpha_k A y_k$$

e.
$$q_{k+1} = M^{-1} r_{k+1}$$
$$w_{k+1} = N^{-1} A^T q_{k+1}$$

f.
$$y_{k+1} = q_{k+1} + \beta_k y_k$$
$$z_{k+1} = w_{k+1} + \beta_k z_k$$

```python
import numpy as np

# define the matrix A
A = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])

# define the right-hand side vector b
b = np.array([1, 0, 1])

# solve the system Ax = b using QMR
max_iter = 1000
tol = 1e-8
x = np.zeros_like(b, dtype=float)
r = b - A @ x
p = r.copy()
q = r.copy()
for k in range(max_iter):
    y = A @ q
    z = A.T @ p
    alpha = np.dot(r, q) / np.dot(y, z)
    beta = np.dot(r, z) / np.dot(y, z)
    x += alpha * p
    r -= alpha * y
    p = r + beta * p
    q = q - beta * z
    if np.linalg.norm(r) < tol:
        print(f"Converged in {k+1} iterations")
        break

# Print the solution
print("Converged in 1000 iterations \n")
print(f"Solution using QMR: {x}")
```

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

example :

$$3x + 2y + z = 6$$
$$2x + 5y + 3z = -12 \quad \Rightarrow \quad A = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 1 & 2 \end{bmatrix}, \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ -12 \\ 4 \end{bmatrix}$$
$$x + y + 2z = 4$$

answer :

1. $x_0 = [0\ 0\ 0]^T \quad , \quad r_0 = b - x_0 = [6\ \ -12\ \ 4]^T$

2. $u, v = I \Rightarrow z_0 = [6\ \ -12\ \ 4]^T \quad , \quad y_0 = [6\ \ -12\ \ 4]^T$

3. for $k = 0, 1, \dots,$ convergence :

a.

$$Ay_0 = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 1 & 2 \end{bmatrix} [6\ \ -12\ \ 4]^T = \begin{bmatrix} -2 \\ -24 \\ 2 \end{bmatrix},$$

$$A^T z_0 = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 3 & 2 \end{bmatrix} [6\ \ -12\ \ 4] = \begin{bmatrix} -14 \\ 44 \\ -22 \end{bmatrix}$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

(b)

$$\alpha_0 = \frac{(r_0, z_0)}{(y_0, Ay_0)} = \frac{196}{284} = 0.69 \quad , \quad \beta_0 = \frac{(r_0, A^T z_0)}{(y_0, A^T z_0)} = \frac{-700}{-700} = 1$$

(c)

$$x_1 = [0\ 0\ 0]^T + 0.69\,[6 \quad -12 \quad 4]^T = [4.14 \quad -8.28 \quad 2.76]^T$$

(d)

$$r_1 = [6 \quad -12 \quad 4]^T - 0.69\,[-2 \quad -24 \quad 2]^T = [7.38 \quad 4.56 \quad 2.62]^T$$

(e)

$$q_1 = [7.38 \quad 4.56 \quad 2.62]^T \quad , \quad w_1 = I \begin{bmatrix} 3 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 3 & 2 \end{bmatrix} \begin{bmatrix} 7.38 \\ 4.56 \\ 2.62 \end{bmatrix} = \begin{bmatrix} 33.88 \\ 40.18 \\ 26.3 \end{bmatrix}$$

(f)

$$y_1 = [7.38 \quad 4.56 \quad 2.62]^T + 1 \times [6 \quad -12 \quad 4]^T = [13.38 \quad 7.44 \quad 6.62]^T$$

$$z_1 = \begin{bmatrix} 33.88 \\ 40.18 \\ 26.3 \end{bmatrix} + 1 \times [6 \quad -12 \quad 4]^T = [39.88 \quad 28.18 \quad 30.3]^T$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EACH ITERATION'S ANSWER (TOL = $1e - 6$)

k	$\frac{\|\|r_k\|\|}{\|\|b\|\|}$
0	1.732
1	0.843
2	0.218
3	0.031
4	0.008
5	0.0003
6	0.00002

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EACH ITERATION'S ANSWER (TOL = $1e-6$) USING IC METHOD FOR PRECONDITIONERS

k	$\frac{\lvert\lvert \boldsymbol{r_k} \rvert\rvert}{\lvert\lvert \boldsymbol{b} \rvert\rvert}$
0	1.732
1	0.196
2	2.018e-08

COMPARISON, NUMBER OF ITERATIONS

QMR without any preconditioners	QMR with preconditioners
7	3

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

TIME AND SPACE COMPLEXITY

time	space
O(n^2)	O(n)

The given time complexity is in general and the actual amount is very dependence on the matrix properties.

STRENGTHS AND WEAKNESSES

STRENGTHS AND WEAKNESSES

STRENGTHS

1. Non-symmetric
2. Indefinite

WEAKNESSES

1. Ill-conditioned
2. Highly singular

NON-SYMMETRIC

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

```
import numpy as np
import time
from scipy.sparse import csc_matrix
from scipy.sparse.linalg import qmr
from scipy.sparse.linalg import cg


# Define the matrix A and vector b
A = csc_matrix([[10, 1, -2], [2, 8, -3], [1, -7, 1]], dtype=float)
b = np.array([1, 1, 1], dtype=float)

start = time.time()
x, exitCode = qmr(A, b)
end = time.time()

print("answer: ", x)
print("exit code: ", exitCode)
print("time: ", end - start)

start = time.time()
x, exit_code = cg(A, b)
end = time.time()

print("answer: ", x)
print("exit code: ", exitCode)
print("time: ", end - start)
```

NON-SYMMETRIC

QMR

```
answer:  [-0.14285714 -0.36263736 -1.3956044 ]
exit code:  0
time:  0.000881195068359375
```

CG

```
answer:  [-0.13971585 -0.36901004 -1.41650883]
exit code:  0
time:  0.000997304916381836
```

ILL-CONDITIONED

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

```python
import numpy as np
import time
from scipy.sparse import csc_matrix
from scipy.sparse.linalg import qmr
from scipy.sparse.linalg import cg


# Define the matrix A and vector b
A = csc_matrix([[1.001, 2], [2, 4.001]], dtype=float)
b = np.array([1, 1], dtype=float)

start = time.time()
x, exitCode = qmr(A, b)
end = time.time()

print("answer: ", x)
print("exit code: ", exitCode)
print("time: ", end - start)

start = time.time()
x, exit_code = cg(A, b)
end = time.time()

print("answer: ", x)
print("exit code: ", exitCode)
print("time: ", end - start)
```

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

ILL-CONDITIONED

QMR

```
answer:  [ 400.119976   -199.76004799]
exit code:  0
time:  0.0011818408966064453
```

CG

```
answer:  [ 400.119976   -199.76004799]
exit code:  0
time:  0.0011818408966064453
```

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OTHER STRENGTHS & WEAKNESSES

STRENGTHS

- Can be implemented in a way that allows for parallelization, hence it is well-suited for large matrices.

WEAKNESSES

- Requires more memory space since it is a bi-orthogonal method.

CONCLUSION

CONCLUSION

- Iterative bi-orthogonal Krylov subspace method.
- Well-suited for indefinite and non-symmetric matrices.
- Can be improved with appropriate preconditioners.
- More precision given it’s a bi-orthogonal method.
- High memory cost.

SOURCES

1. Yousef Saad "Iterative Methods for Sparse Linear Systems”
2. https://mathworld.wolfram.com/Quasi-MinimalResidualMethod.html
3. C.T.Kelley"Iterativemethodsforlinearandnonlinearequations"

THANKS FOR YOUR ATTENTION. CONTACT INFO:

sarinaheshmatii@gmail.com

sarinaheshmati@aut.ac.ir

+989102460274