Conjugate Gradient Method

Large scaled linear system

The Gram–Schmidt process

Input and Output

- Input: linearly independent set of vectors in $\mathbb{R}^n$:
  $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}, k \leq n$

- Output: generate an orthogonal set, $S' = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$

- S' spans the same $k - dimensional$ subspace of $\mathbb{R}^n$ as S

Gram-Schmidt algorithm

$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}
$$

```
function U = gramschmidt(V)
    [n, k] = size(V);
    U = zeros(n,k);
    U(:,1) = V(:,1) / norm(V(:,1));
    for i = 2:k
        U(:,i) = V(:,i);
        for j = 1:i-1
            U(:,i) = U(:,i) - (U(:,j)'*U(:,i)) * U(:,j);
        end
        U(:,i) = U(:,i) / norm(U(:,i));
    end
end
```

```
V = [3 2;1 2]
U = gramschmidt(V)
U =
```

```
0.9487   -0.3162
0.3162    0.9487
```

Solve linear system

- Solve linear equation $\boldsymbol{Ax} = \boldsymbol{b}$

  $(or\ min\ f(x) = \frac{1}{2}x^T Ax - x^T b\ )$

where A is n-by-n **symmetric** matrix (i.e. $A^T = A$)

positive definite (i.e. $x^T Ax > 0$ for all x in R^n)

Let $x_* = A^{-1}b$ denote the exact solution, and let $e_k \equiv x_* - x_k$ denote the error at step k

Also, let $\|\cdot\|_A$ denote the norm

$$\|x\|_A \equiv \sqrt{x^T A x}$$

Show by induction

- Let $u_0 = v_0$
- $u_k = v_k - \sum_{j<k} \frac{u_j^t A v_k}{u_j^t A u_j} u_j$ for k = 1, 2,…
- $u_j^t A u_, = 0, 0 \leq m < j \leq k$
- Proof : by induction!
- When k = 1, $u_1 = v_1 - \frac{u_0^t A v_1}{u_0^t A u_0} u_0$
- Assume when k = i, $u_1, \ldots, u_i$ are pairwise conjugate
- We need to show $u_{i+1} A u_m = 0$, for all $m \leq i$

The CG algorithm

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$

if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result

$\mathbf{p}_0 := \mathbf{r}_0$

$k := 0$

repeat

$$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T} \mathbf{r}_k}{\mathbf{p}_k^\mathsf{T} \mathbf{A} \mathbf{p}_k}$$

EQ 1

$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$

EQ 2

$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$

EQ 3

if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop

$$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T} \mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T} \mathbf{r}_k}$$

EQ 5

$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

EQ 4

$k := k + 1$

end repeat

return $\mathbf{x}_{k+1}$ as the result

Show pairwise conjugate by induction

- Let $p_0 = r_0$
- $p_k = r_k - \sum_{j<k} \frac{p_j^t A r_k}{p_j^t A p_j} p_j$ for k = 1, 2,…
- $p_j^t A p_m = 0, 0 \le m < j \le k$
- Proof : by induction!
- When k = 1, since $p_1 = r_1 - \frac{p_0^t A r_1}{p_0^t A p_0} p_0$, $p_1^t A p_0 = r_1^t A p_0 - \frac{p_0^t A r_1}{p_0^t A p_0} p_0^t A p_0 = 0$
- Assume when k = i, $p_1, \ldots, p_i$ are pairwise conjugate
- We need to show $p_{i+1} A p_m = 0$, for all $m \le i$

Theorem: The CG iterate x_k is the unique member of $\mathcal{K}_k(A, b)$ that minimizes $\|e_k\|_A$. Also, $x_k = x_*$ for some $k \leq n$.

Proof: This result relies on a set of identities that can be derived (by induction) from the CG algorithm:

(i) $\mathcal{K}_k(A, b) = \text{span}\{x_1, x_2, \ldots, x_k\} =$
$\text{span}\{p_0, p_1 \ldots, p_{k-1}\}$
$= \text{span}\{r_0, r_1, \ldots, r_{k-1}\}$

(ii) $r_k^T r_j = 0$ for $j < k$

(iii) $p_k^T A p_j = 0$ for $j < k$

Derivation I

Let $x_k = x_{k-1} + \alpha_k p_k, \quad x_k = \sum_{i=1}^{k} \alpha_i p_i$

EQ 2

$$\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{p}_i \Rightarrow \mathbf{A}\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{A}\mathbf{p}_i .$$

$$\mathbf{p}_k^{\mathsf{T}} \mathbf{b} = \mathbf{p}_k^{\mathsf{T}} \mathbf{A}\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{p}_k^{\mathsf{T}} \mathbf{A}\mathbf{p}_i = \sum_{i=1}^{n} \alpha_i \langle \mathbf{p}_k, \mathbf{p}_i \rangle_{\mathbf{A}} = \alpha_k \langle \mathbf{p}_k, \mathbf{p}_k \rangle_{\mathbf{A}}$$

and so

$$\alpha_k = \frac{\langle \mathbf{p}_k, \mathbf{b} \rangle}{\langle \mathbf{p}_k, \mathbf{p}_k \rangle_{\mathbf{A}}}.$$

Derivation II

Let $\mathbf{r}_k$ be the residual at the kth step:

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k.$$

Current negative gradient

Recursive Form :

$$r_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k p_k)$$
$$= b - Ax_k - \alpha_k Ap_k = r_k - \alpha_k Ap_k$$

EQ 3

Derivation III

Let $p_{k+1} = r_{k+1} + \beta_k p_k$ EQ 4

$$p_{k+1}^T A p_k = r_{k+1}^T A p_k + \beta_k p_k^T A p_k$$

$$\beta_k = -\frac{\mathbf{r}_{k+1}^\mathsf{T} \mathbf{A} \mathbf{p}_k}{\mathbf{p}_k^\mathsf{T} \mathbf{A} \mathbf{p}_k}$$

$$\beta_k = -\frac{\mathbf{r}_{k+1}^\mathsf{T} \mathbf{A}\mathbf{p}_k}{\mathbf{p}_k^\mathsf{T} \mathbf{A}\mathbf{p}_k}$$

Nominator

denominator

using

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$$

and equivalently

$$\mathbf{A}\mathbf{p}_k = \frac{1}{\alpha_k}(\mathbf{r}_k - \mathbf{r}_{k+1}),$$

the numerator of β_k is rewritten as

$$\mathbf{r}_{k+1}^\mathsf{T} \mathbf{A}\mathbf{p}_k = \frac{1}{\alpha_k}\mathbf{r}_{k+1}^\mathsf{T}(\mathbf{r}_k - \mathbf{r}_{k+1}) = -\frac{1}{\alpha_k}\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}$$

because $\mathbf{r}_{k+1}$ and $\mathbf{r}_k$ are orthogonal by design. The denominator is rewritten as

$$\mathbf{p}_k^\mathsf{T} \mathbf{A}\mathbf{p}_k = (\mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1})^\mathsf{T} \mathbf{A}\mathbf{p}_k = \frac{1}{\alpha_k}\mathbf{r}_k^\mathsf{T}(\mathbf{r}_k - \mathbf{r}_{k+1}) = \frac{1}{\alpha_k}\mathbf{r}_k^\mathsf{T}\mathbf{r}_k$$

$= r_k^t A p_k$

$$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T} \mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T} \mathbf{r}_k}$$

EQ 5

because $\mathbf{r}_{k+1}$ and $\mathbf{r}_k$ are orthogonal by design. The denominator is rewritten as

$$\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k = (\mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1})^\mathsf{T}\mathbf{A}\mathbf{p}_k = \frac{1}{\alpha_k}\mathbf{r}_k^\mathsf{T}(\mathbf{r}_k - \mathbf{r}_{k+1}) = \frac{1}{\alpha_k}\mathbf{r}_k^\mathsf{T}\mathbf{r}_k$$

$$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$$

EQ 1

Conjugate gradient method

14 languages

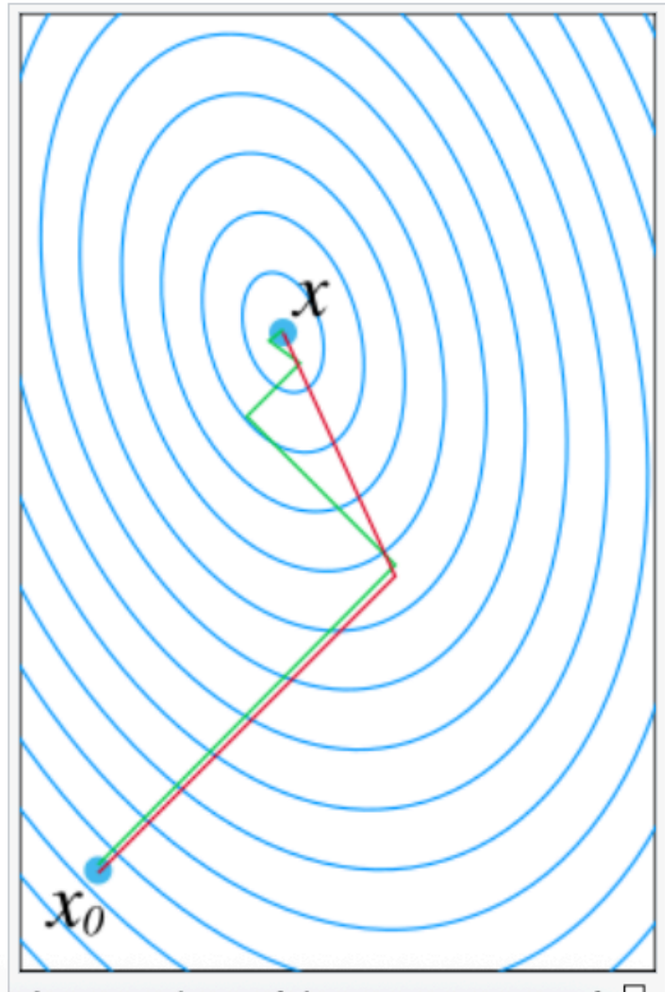
Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

In mathematics, the **conjugate gradient method** is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is positive-semidefinite. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to ...her direct methods such as the Cholesky decomposi... ...numerically solving partial differential equations ...

The conjugate gradient met... ...optimization problems such as energy minimizatio... ...es and Eduard Stiefel,[1] [2] who programmed it on th...

The biconjugate gradient method provides a generalization to non-symmetric matrices. Various nonlinear conjugate gradient methods seek minima of nonlinear optimization problems.



A comparison of the convergence of

In computational mathematics, an **iterative method** is a mathematical procedure that uses an initial value to generate a sequence of improving approximate solutions for a class of problems, in which the *i*-th approximation is derived from the previous ones.

Description of the problem addressed by conjugate gradients [edit]

Suppose we want to solve the system of linear equations

/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/conjugate_gradient.m

```m
function x = conjugate_gradient(A, b, x0, tol)
    if nargin < 4
        tol = eps;
    end
    r = b - A * x0;
    p = r;
    rsold = r' * r;
    x = x0;
    while sqrt(rsold) > tol
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end
```

```
>> x = conjugate_gradient(A, b, x0)

x =

    0.0909
    0.6364
```

The conjugate gradient method aims to solve a system of linear equations, Ax=b, where A is symmetric, without calculation of the inverse of A. It only requires a very small amount of membory, hence is particularly suitable for large scale systems.

It is faster than other approach such as Gaussian elimination if A is well-conditioned. For example,

```
n=1000;
[U,S,V]=svd(randn(n));
s=diag(S);
A=U*diag(s+max(s))*U'; % to make A symmetric, well-contioned
b=randn(1000,1);
tic,x=conjgrad(A,b);toc
tic,x1=A\b;toc
norm(x-x1)
norm(x-A*b)
```

Experiment I. Left division

/ > Users > jiann-mingwu > Desktop > JiannMingWu > course2025-online > NA2025 > conjugate Gradient >

gramschmidt.m × | conjugate_gradient.m × | demo_cgMethod.m × | CGSLinearSystemExample.mlx × | conjgrad.m ×

/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/demo_cgMethod.m

```
5       x = randn(n,1);
6       b = A * x;
7       select = 1;
8       fprintf('size A : %d x %d, size b : %d \n', size(A,1),siz
9       switch select
10          case 1
11              fprintf('solve by left div \n')
12              tic
```

Command Window

```
>> demo_cgMethod
size A : 6000 x 6000, size b : 6000
solve by left div
exe time 0.457381, mean_abs_error 0.000000
```

Experiment II

/ > Users > jiann-mingwu > Desktop > JiannMingWu > course2025-online > NA2025 > conjugate Gradient >

gramschmidt.m × | conjugate_gradient.m × | demo_cgMethod.m × | CGSLinearSystemExample.mlx × | conjgrad.m ×

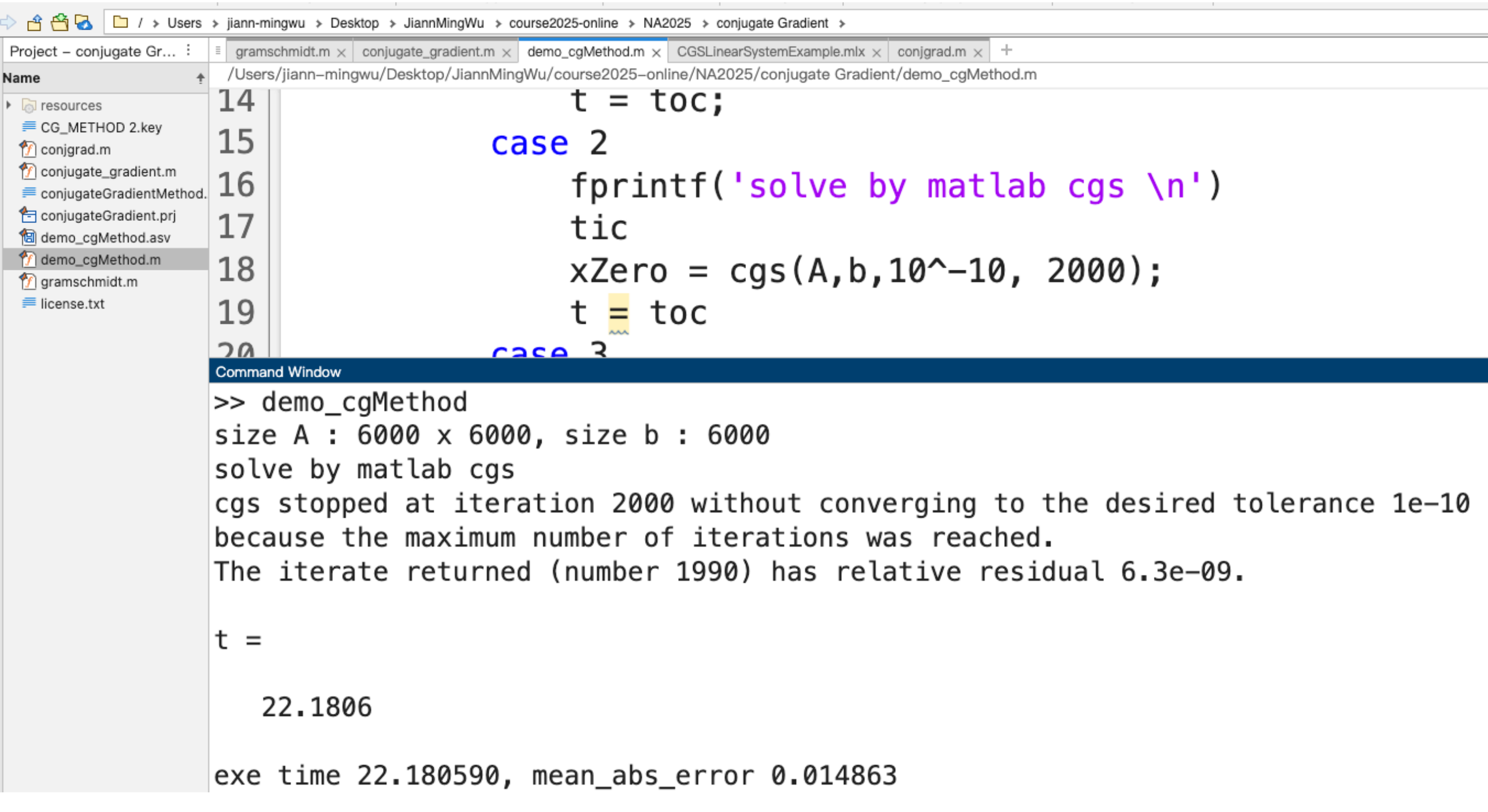
/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/demo_cgMethod.m

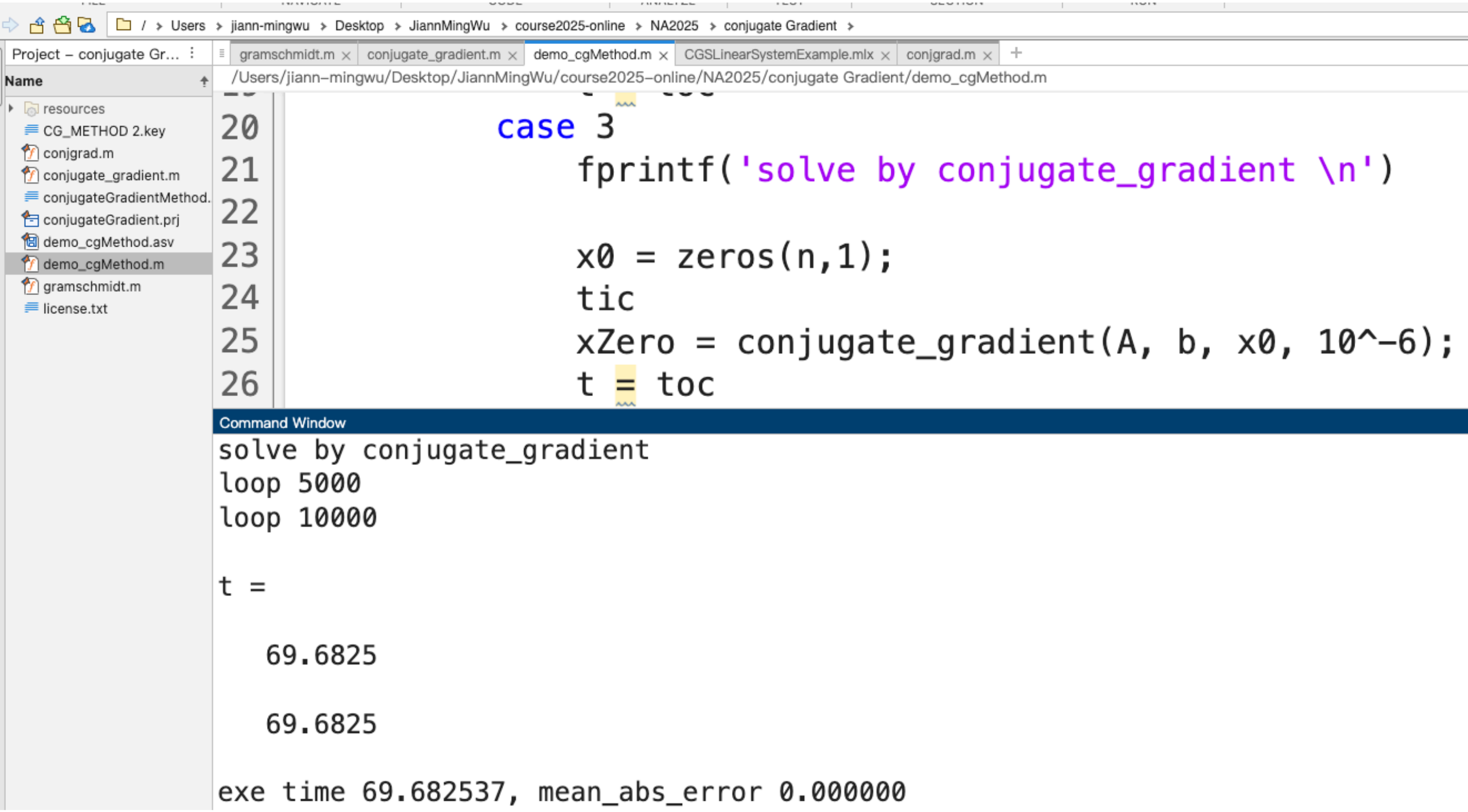
```
13              xZero = A\b;
14              t = toc;
15          case 2
16              fprintf('solve by matlab cgs \n')
17              tic
18              xZero = cgs(A,b,10^-6, 200);
19              t = toc
20
```

Command Window

```
t =

    2.1455

exe time 2.145545, mean_abs_error 0.050798
```

```
>> demo_cgMethod
size A : 6000 x 6000, size b : 6000
solve by matlab cgs
cgs stopped at iteration 200 without converging to the desired tolerance 1e-06
because the maximum number of iterations was reached.
The iterate returned (number 200) has relative residual 2e-06.

t =

    2.1455

exe time 2.145545, mean_abs_error 0.050798
```


/ > Users > jiann-mingwu > Desktop > JiannMingWu > course2025-online > NA2025 > conjugate Gradient >
Project – conjugate Gr...
Name
resources
CG_METHOD 2.key
conjgrad.m
conjugate_gradient.m
conjugateGradientMethod.
conjugateGradient.prj
demo_cgMethod.asv
demo_cgMethod.m
gramschmidt.m
license.txt
gramschmidt.m
conjugate_gradient.m
demo_cgMethod.m
CGSLinearSystemExample.mlx
conjgrad.m
/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/demo_cgMethod.m
14 t = toc;
15 case 2
16 fprintf('solve by matlab cgs \n')
17 tic
18 xZero = cgs(A,b,10^-10, 2000);
19 t = toc
20 case 3
Command Window
>> demo_cgMethod
size A : 6000 x 6000, size b : 6000
solve by matlab cgs
cgs stopped at iteration 2000 without converging to the desired tolerance 1e-10
because the maximum number of iterations was reached.
The iterate returned (number 1990) has relative residual 6.3e-09.
t =
22.1806
exe time 22.180590, mean_abs_error 0.014863


/ > Users > jiann-mingwu > Desktop > JiannMingWu > course2025-online > NA2025 > conjugate Gradient >
Project – conjugate Gr...
Name
resources
CG_METHOD 2.key
conjgrad.m
conjugate_gradient.m
conjugateGradientMethod.
conjugateGradient.prj
demo_cgMethod.asv
demo_cgMethod.m
gramschmidt.m
license.txt
gramschmidt.m
conjugate_gradient.m
demo_cgMethod.m
CGSLinearSystemExample.mlx
conjgrad.m
/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/demo_cgMethod.m
20 case 3
21 fprintf('solve by conjugate_gradient \n')
22
23 x0 = zeros(n,1);
24 tic
25 xZero = conjugate_gradient(A, b, x0, 10^-6);
26 t = toc
Command Window
solve by conjugate_gradient
loop 5000
loop 10000
t =
69.6825
69.6825
exe time 69.682537, mean_abs_error 0.000000

gramschmidt.m | conjugate_gradient.m | demo_cgMethod.m | CGSLinearSystemExample.mlx | conjgrad.m

/Users/jiann-mingwu/Desktop/JiannMingWu/course2025-online/NA2025/conjugate Gradient/demo_cgMethod.m

```
27              disp(t)
28          otherwise
29              tic
30              xZero=conjgrad(A,b);
31              t = toc
32      end
```

Command Window

```
>> demo_cgMethod
size A : 6000 x 6000, size b : 6000

t =

   33.4071

exe time 33.407133, mean_abs_error 0.003303
```

```
>> demo_cgMethod
size A : 20000 x 20000, size b : 20000
solve by left div
exe time 22.527196, mean_abs_error 0.000000
>> demo_cgMethod
size A : 20000 x 20000, size b : 20000
solve by matlab cgs
cgs stopped at iteration 2000 without converging to the desired tolerance 1e-10
because the maximum number of iterations was reached.
The iterate returned (number 1702) has relative residual 4.6e-09.

t =

  192.3454

exe time 192.345371, mean_abs_error 0.001301
```

```
>> demo_cgMethod
size A : 25000 x 25000, size b : 25000
solve by matlab cgs
cgs converged at iteration 266 to a solution with relative residual 1e-06.

t =

   49.3257

exe time 49.325651, mean_abs_error 0.042350
```

Command Window

```
>> demo_cgMethod
size A : 30000 x 30000, size b : 30000
solve by matlab cgs
cgs converged at iteration 273 to a solution with relative residual 1e-06.

t =

  154.0462

exe time 154.046192, mean_abs_error 0.038663
```

```
>> demo_cgMethod
size A : 35000 x 35000, size b : 35000
solve by matlab cgs
cgs stopped at iteration 2000 without converging to the desired tolerance 1e-06
because the maximum number of iterations was reached.
The iterate returned (number 2000) has relative residual NaN.

t =

   1.3710e+03

exe time 1371.023696, mean_abs_error NaN
```