WROCLAW UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRONICS

FIELD: Electronics
SPECIALITY: Advanced Applied Electronics

Numerical Methods: Eigenproblems

AUTHOR:
Jaroslaw M. Szumega

SUPERVISOR:
Rafal Zdunek, D.Sc, K-4/W4

GRADE:

WROCŁAW 21.03.2017

Contents

Chapter 1

Solution to the given problems

1.1 Problem 1

Problem 1: Find the solution that best approximates the system of inconsistent linear equations:

$$\text{(a)} \begin{cases} 3x_1 - x_2 = 4 \\ x_1 + 2x_2 = 0 \\ 2x_1 + x_2 = 1 \end{cases}, \qquad \text{(b),} \begin{cases} 3x_1 + x_2 + x_3 = 6 \\ 2x_1 + 3x_2 - x_3 = 1 \\ 2x_1 - x_2 + x_3 = 0 \\ 3x_1 - 3x_2 + 3x_3 = 8 \end{cases}, \qquad \text{(c)} \begin{cases} x_1 + x_2 - x_3 = 5 \\ 2x_1 - x_2 + 6x_3 = 1 \\ -x_1 + 4x_2 + x_3 = 0 \\ 3x_1 + 2x_2 - x_3 = 6 \end{cases}.$$

1.2 Problem 2

1.3 Problem 3

1.4 Problem 4

1.5 Problem 5

1.6 Problem 6

1.7 Problem 7

1.8 Problem 8

1.9 Problem 9

1.10 Problem 10

1.11 Problem 11

Chapter 2

Algorithms code

Algorithm 1 – The classical LS fitting.

```
function [x] = classicLS(A, b)

[m,n] = size(A)

if(m >= n || rank(A) == n)
    disp(["There is an unique solution"])
    x = inv(A' * A) * A' * b;
else if ( m < n)
    disp(["Underdetermined system"])
    x = A' * inv(A * A') * b;
end
endfunction
```

Algorithm 2 – Pseudoinverse

```
function [x] = pseudoinverse(A)

[m,n] = size(A);
r = rank(A);
[u,s,v] = svdqr(A,20);

x = zeros(n,m);

for i = 1:r
    x = x + inv(s(i,i)) *v(:,i)* u(:,i)';
endfor

endfunction
```

Algorithm 3 – The orthogonal projectors

```
function [Pra, Prah, Pna, Pnah] = projectors(A)

[m,n] = size(A)

Pra = A * pseudoinverse(A)
Prah = pseudoinverse(A) * A

Pnah = eye(size(Pra)) - Pra
Pna = eye(size(Prah)) - Prah

endfunction
```

Algorithm 4 – The LS solution by SVD

```
function [x] = svdLS(A,b)

[u,s,v] = svd(A);

x = (v*pseudoinverse(s) * u') * b;

endfunctiong
```

Algorithm 5 – The LS solution by QR factorization

```
function [U, S, V] = svdQR(A,iterations)

[n, m]= size(A); # can be rectangular matrix
U=eye(n);
V=eye(m);

R=A';

for i = 0:iterations
    [Q,R]=qr(R');   # qr decompositions and updating
    U=U*Q;
    [Q,R]=qr(R');
    V=V*Q;
endfor
S=R';               # S is R transposed

endfunction
```

Algorithm 6 – The linear regression

```
function [x] = regression(A, b)

[m,n] = size(A
[p,r] = size(b);

if(n != 2 || r != 1)
    disp(["The matrix does not describe the polynomial of first degree"
       ])

else
    meanY = sum(b)/p
    meanT = sum(A(:,n))/m

    #beta = (sum(b.*A(:,n)) - m*meanY*meanT)/(sum(A(:,n).^2) - m * meanT
       *meanT)
    #alpha = meanY - beta*meanT

    #more accurate b calculation
    first = (b.- meanY).*(A(:,n).-meanT)
    second = (b.-meanT).^2
    beta = sum(first)/sum(second)
    alpha = meanY - beta*meanT

    x = [alpha;beta]
endif
endfunction
```

Algorithm 7 – The TSVD algorithm

```
function [x] = tsvd(A,b, it)

[u,s,v] = svdqr(A,it);

c = u'*b
r = rank(A)
x=zeros(size(A,2),1);

for i = 1:r
    x = x + (c(i)*v(:,i))/s(i,i);
endfor
endfunction
```

Algorithm 8 – The iterative refinement

```
function [x] = refinement(A,x,b,it)

for s = 1:it
    r = b - A*x

    #extended refinement
    delta = qrLS(A,r);
    x = x + delta
endfor
endfunction
```

Algorithm 10 – The General Cross-Validation

```
function [x] = crossvalidation(A,b, mi)

C = inv(A'*A);
M = A' * A + (mi.^2).^C'*C;

x = inv(M)*A'*b;

endfunction
```

Algorithm 11 – The Iterative Tikhonov Regularization

```
function [x] = tikhonovIt(A,b, it, mi)

[m,n] = size(A);

x=zeros(size(A,2),1);
for i = 1:it
    x = x + inv(A' * A + eye(size(A'*A)).*mi) * A' * (b-A*x);
endfor

endfunction
```

Algorithm ** – The General Tikhonov Regularization

```
function [x] = tikhonovGen(A,b, alpha)

x = inv(A' * A + alpha.*eye(size(A'*A))) * A' * b;

endfunction
```

Bibliography

[1] Björck, Åke. Numerical methods for least squares problems. Society for Industrial and Applied Mathematics, 1996.

[2] Golub, Gene H., and Charles F. Van Loan. "Matrix computations, 3rd." (1996).

[3] Zdunek R., Numerical Methods - lecture slides.