

```
1 # Solving Bevington example 6.1 with Octave
2
3 printf( "Bevington Example 6.1\n" )
4 strftime ( "%Y-%m-%d %H:%M:%S", localtime (time ()))
5
6 # design matrix
7 printf( "design matrix:\n" )
8 A = [ 1 1; 1 2; 1 3; 1 4; 1 5; 1 6; 1 7; 1 8; 1 9 ]
9
10 # Define data
11 printf( "data vector:\n" )
12 T = [15.6; 17.5; 36.6; 43.8; 58.2; 61.6; 64.2; 70.4; 98.8]
13
14 # solve least squares problem
15 printf( "least squares solution\n" )
16 printf( "xls = A \ T:\n" )
17 xls = A \ T
18
19 # residual error vector
20 residual = A * xls - T;
21 printf( "residual error vector = A * xls - T\n" )
22
23 t2 = dot( residual, residual );
24 printf( "least total squared error t2 = residual . residual = %d\n", t2 )
25
26 # compute Gram matrix
27 W = transpose( A ) * A;
28
29 # invert Gram matrix
30 Winv = inv( W );
31 values = diag( Winv );
32
33 # measure design matrix
34 # m = rows, n = columns
35 [ m, n ] = size ( A );
36 g = sprintf( '%d ', [ m, n ] );
37 fprintf( 'matrix dimensions: %s\n', g );
38
39 printf( "\ncompute error elements:\n" )
40 printf( "sigma = sqrt( t2 / ( m - n ) * values ):\n" )
41 sigma = sqrt( t2 / ( m - n ) * values )
42
43 printf( "\n# # # Compare Octave values to exact values\n" )
44 printf( "\nFit parameters\n" )
45
46 printf( "\nerror in intercept and slope values\n" )
47 printf( "numericError = xls - [ 1733 / 360; 1129 / 120 ]\n" )
48 numericError = xls - [ 1733 / 360; 1129 / 120 ]
```

```
49
50 printf( "\nerror in intercept and slope values in machine epsilon\n" )
51 printf( "epsError = numericError ./ eps( 1.0 )\n" )
52 epsError = numericError ./ eps( 1.0 )
53
54 printf( "\nError parameters\n" )
55 printf( "\nintercept and slope sigmas\n" )
56 printf( "numericError = sigma - sqrt( [ 108297055; 3419907 ] / 35 ) /
... 360\n" )
57 numericError = sigma - sqrt( [ 108297055; 3419907 ] / 35 ) / 360
58
59 printf( "\nerror in intercept and slope sigmas in machine epsilon\n" )
60 printf( "epsError = numericError ./ eps( 1.0 )\n" )
61 epsError = numericError ./ eps( 1.0 )
62
63 ## dantopa@Quaxolotl.local:least-squares $ pwd
64 ## /Volumes/T7-Touch/repos/github/jop/octave/genesis/least-squares
65 ## dantopa@Quaxolotl.local:least-squares $ octave-cli wtf.m
66 ## Bevington Example 6.1
67 ## ans = 2022-09-06 20:41:57
68 ## design matrix:
69 ## A =
70
71 ##      1      1
72 ##      1      2
73 ##      1      3
74 ##      1      4
75 ##      1      5
76 ##      1      6
77 ##      1      7
78 ##      1      8
79 ##      1      9
80
81 ## data vector:
82 ## T =
83
84 ##      15.600
85 ##      17.500
86 ##      36.600
87 ##      43.800
88 ##      58.200
89 ##      61.600
90 ##      64.200
91 ##      70.400
92 ##      98.800
93
94 ## least squares solution
95 ## xls = A \ T:
```

```
96 ## xls =
97
98 ##      4.8139
99 ##      9.4083
100
101 ## residual error vector = A * xls - T
102 ## least total squared error t2 = residual . residual = 316.658
103 ## matrix dimensions: 9 2
104
105 ## compute error elements:
106 ## sigma = sqrt( t2 / ( m - n ) * values ):
107 ## sigma =
108
109 ##      4.8862
110 ##      0.8683
111
112
113 ## # # # Compare Octave values to exact values
114
115 ## Fit parameters
116
117 ## error in intercept and slope values
118 ## numericError = xls - [ 1733 / 360; 1129 / 120 ]
119 ## numericError =
120
121 ##      -1.0658e-14
122 ##      1.7764e-15
123
124
125 ## error in intercept and slope values in machine epsilon
126 ## epsError = numericError ./ eps( 1.0 )
127 ## epsError =
128
129 ##      -48
130 ##      8
131
132
133 ## Error parameters
134
135 ## intercept and slope sigmas
136 ## numericError = sigma - sqrt( [ 108297055; 3419907 ] / 35 ) / 360
137 ## numericError =
138
139 ##      8.8818e-16
140 ##      1.1102e-16
141
142
143 ## error in intercept and slope sigmas in machine epsilon
```

```
144 ## epsError = numericError ./ eps( 1.0 )
145 ## epsError =
146
147 ##      4.0000
148 ##      0.5000
149
```