1.20

Solution.

(a) The derivative of J with respect to c_1 is

$$\frac{2}{(1+c_2^2)}(c_1 + c_2 m_a - m_b).$$

Setting this to zero gives $c_1 = m_b - m_a c_2$.

(b) Expanding the square in the numerator gives

$$\begin{aligned} & \|c_2(a - m_a\mathbf{1}) - (b - m_b)\mathbf{1}\|^2 \\ &= c_2^2\|a - m_a\mathbf{1}\|^2 + \|b - m_b\mathbf{1}\|^2 - 2c_2(a - m_a\mathbf{1})^T(b - m_b\mathbf{1}) \\ &= n\left(c_2^2 s_a^2 + s_b^2 - 2\rho c_2 s_a s_b\right). \end{aligned}$$

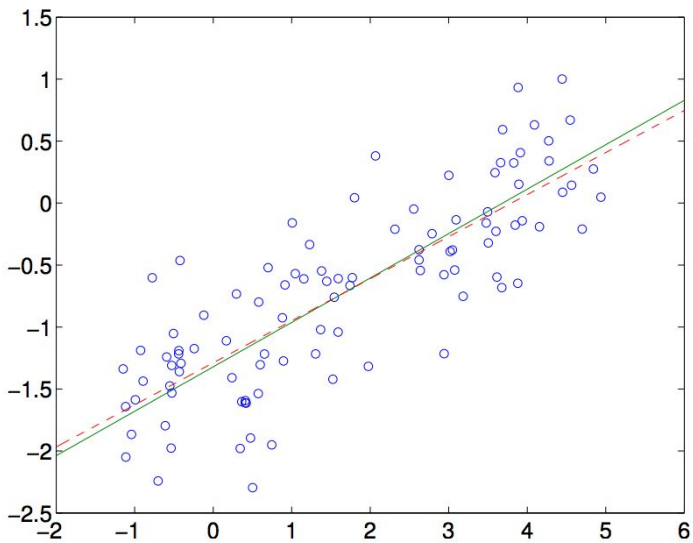
Setting the derivative to zero gives an equation

$$\frac{2s_a^2 c_2 - 2\rho s_a s_b}{1 + c_2^2} - \frac{2c_2(s_a^2 c_2^2 + s_b^2 - 2\rho s_a s_b c_2)}{(1 + c_2^2)^2} = 0.$$

After simplifications this reduces to the quadratic equation.

To see that the root with the same sign as ρ is the correct one, note that J approaches a limit s_a^2 as $c_2 \to \infty$ and $c_2 \to -\infty$. Also there is a positive and a negative root. One of these roots must correspond to a maximum of J, and the other to a minimum. If $\rho > 0$ the derivative of J at $c_2 = 0$ is negative. Therefore the positive root is a minimum. If $\rho < 0$ the derivative of J at $c_2 = 0$ is positive, and the negative root is a minimum.

(c) The plot shows the two results. The solid line is the orthogonal distance regression fit. The dashed line is the least-squares fit.



The least-squares solution is $c_1 = -1.2891$ and $c_2 = 0.3393$. The orthogonal distance regression gives $c_1 = -1.3214$ and $c_2 = 0.3584$.

1.21

Solution. The correct answer is (b). This can be seen by elimination. The two hyperplanes H_1 and H_2 do not change if we multiply the vector a and the scalars b and c by the same nonzero number: the equations

$$(\lambda a)^T x = \lambda b, \qquad (\lambda a)^T y = \lambda c,$$

with $\lambda \neq 0$, define the same hyperplanes. Therefore (a) and (c) are wrong because the distance should be invariant with respect to such a scaling.

To prove that (b) is correct, we can use a similar method as in lecture 2, page 35. All points in H_1 can be written as

$$x = \frac{b}{\|a\|^2} a + u$$

with $u^T a = 0$. All points in H_2 can be written as

$$y = \frac{c}{\|a\|^2} a + v$$

with $v^T a = 0$. The squared distance between the two is

$$\begin{aligned} \|x - y\|^2 &= \|\frac{b-c}{\|a\|^2} a + u - v\|^2 \\ &= \frac{|b-c|^2}{\|a\|^2} + \|u - v\|^2 + 2\frac{b-c}{\|a\|^2} a^T (u - v) \\ &= \frac{(b-c)^2}{\|a\|^2} + \|u - v\|^2. \end{aligned}$$

(The last line follows from $a^T u = a^T v = 0$.) We see that the squared distance is minimized by taking $u = v$. The minimal value is

$$d^2 = \frac{(b-c)^2}{\|a\|^2}.$$

1.23

Solution. The MATLAB code is as follows.

```
load tomography
[m,n] = size(A);
x = zeros(n,1);
for k=1:10
    for i =  1:m
        x = x - ( (A(i,:)*x - b(i)) / norm(A(i,:))^2 ) * A(i,:)';
    end;
    disp(sprintf('residual = %e.', norm(b-A*x)/norm(b)));
    subplot(2,5,k);
    imshow(reshape(x, 28, 28));
end;
```

The printed residuals are

```
residual = 1.083536e-01.
residual = 4.356205e-02.
residual = 2.954238e-02.
residual = 2.439667e-02.
```

```
residual = 2.124433e-02.
residual = 1.894906e-02.
residual = 1.716171e-02.
residual = 1.570550e-02.
residual = 1.448036e-02.
residual = 1.342670e-02.
```

The figure shows the reconstructed images for the first ten iterations.

2.13

- $y = (I + uv^T)x$. We first compute uv^T (n^2 flops, as in the previous problem). Then we add I (n flops because we add one to each diagonal element). Finally we multiply the result with x ($2n^2$ flops). The total is $3n^2 + n$ flops.
- $y = x + (v^T x)u$. We first compute $v^T x$ ($2n$ flops). The result is a scalar, so multiplying with x costs n. Then we add x (n flops). The total is $4n$ flops.

On a 900Mhz Pentium III with 256MB of memory, we get the following CPU times

- For $n = 1000$, $t_1 = 0.31$ seconds, $t_2 = 0$ seconds.
- For $n = 2000$, $t_1 = 1.29$ seconds, $t_2 = 0$ seconds.

This confirms that method 2 is faster. For the first method the execution time increases roughly by a factor of four if we double n, which is what we expect. The second method is so fast that the execution time is negligible for $n = 2000$. If we try the second method for $n = 10000$, and $n = 100000$ we get $t_2 = 0.03$ seconds and $t_2 = 0.27$ seconds, *i.e.*, an increase by a factor of about 10, which is what we expect.

Obviously, your results will be different depending on your computer (CPU and memory). Also note that first method requires much more memory. We calculate an $n \times n$ matrix as an intermediate result, which requires $8n^2$ bytes of memory (8 bytes per floating-point number). If that exceeds the amount of available RAM, the computer slows down considerably because it starts using the hard disk as extra memory.

2.14

Solution. $2n^2$ flops. The trace of AB is $\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ji}$. This requires n^2 multiplications and $n^2 - 1$ additions.

4.3

Solution. The rows of a node-arc incidence matrix are linearly dependent. Each colum contains one element equal to 1, one element equal to -1, and the other elements are zero, so the sum of the rows are zero ($\mathbf{1}^T A = 0$). A node-arc incidence matrix is therefore never right invertible (lecture 4, page 26).

To see this directly, if a right inverse X existed, we would have the following contradiction. On the one hand, $\mathbf{1}^T AX = 0$ because $\mathbf{1}^T A = 0$. On the other hand, $\mathbf{1}^T AX = \mathbf{1}^T$ because $AX = I$.