Danfoss: Optimizing control procedure in a mass-production line

C. Henriksen[*], P.G. Hjorth[†], J. Starke [‡]
M. Willatzen[§], A. Sierakowski[¶], H. Schultz [‖]

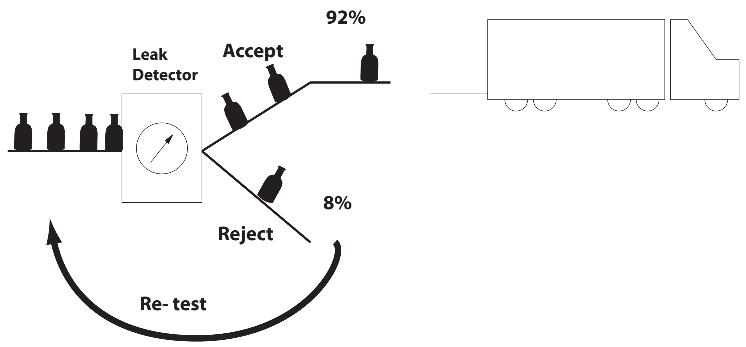
Danfoss representative: P. Johannesen

[*]Technical University of Denmark.
[†]Technical University of Denmark.
[‡]Technical University of Denmark.
[§]University of Southern Denmark.
[¶]University of Southern Denmark.
[‖]University of Southern Denmark.

1 Introduction: The Problem

Consider some units (for the purpose of this report taken to be bottles) that are mass produced on a fully automated assembly line. A property Q of the product (leak rate of He under high pressure) is checked for every unit by a leak detector. Based on the Q-reading of the leak detector, a decision is made to either ship the unit or to re-examine it. If the Q value of the bottle is below a certain value, the bottle is shipped. If the Q-value of the bottle is too high, the bottle fails and is rejected.

With the current control procedure, about 92% of units initially pass the check and are shipped, whereas about 8% fail and are then sent back for re-examination. This is because the Q measurement is noisy and may reject good bottles (whereas it is not prone to accept bad bottles, see section 5.2).



Of the 8% re-tested, almost all eventually pass the inspection; only about .4% of the original bottles are ‘truly’ bad, i.e., have a ‘true’ Q value exceeding specifications.

With about 2.5 million units produced annualy, the 8% initial rejection rate and subsequent re-examination is costly, and presumably unnecessary.

The question for the study group is how to bring down this number of ‘false negative’ decisions (good bottles rejected), without introducing many ‘false positives’ (bad bottles accepted), and still maintain production line speed. Only a few seconds of measuring time is granted each unit.

The decision to reject or accept is based on a comparison of the $Q(t)$ to a certain threshold value Q_T. If the highest registred Q-value Q_{max} is greater than Q_T, the bottle is rejected (re-examined). If the If the highest registred Q-value Q_{max} is smaller than Q_T, the bottle is accepted.

2 Sampling?

The Study Group first considered the question whether a revised sampling procedure might be able to change the situation.

Any sampling algorithm however is based on the available data, i.e., the assignment of a Q value to each unit. The assignment of Q values sets up a probability distribution, which sampling algorithms can then probe in various ways.

It is impossible for any sampling algorithm to circumvent an erroneous assignment of Q values. In other words, clever sampling may save effort in establishing a given probability distribution; but a sampling cannot 'repair' en erroneous probability distribution.

The Study Group therefore decided to focus instead on a more sophisticated processing of the signal $Q(t)$ from the measuring device, aiming to create (within the constraint of a rapid measuring procedure) a more correct assignment of Q-values to bottles.

We begin by modelling the physics of leak detection.

3 Leak Detection Model

Consider a pressure-driven system consisting of two chambers with pressures P_1 and P_2 (denoted C_1 and C_2), respectively (see figure 1). The two chambers are separated by a small hole allowing for particle (He) transport. A pipe with radius R connected to C_2 allows particle flow to the detector where vacuum conditions are assumed to exist. The detector is located at a distance L from C_2.

The rate equation describing the build-up of helium atoms in C_2 is:

$$\frac{dN_2}{dt} = F_{in} - F_{out},$$

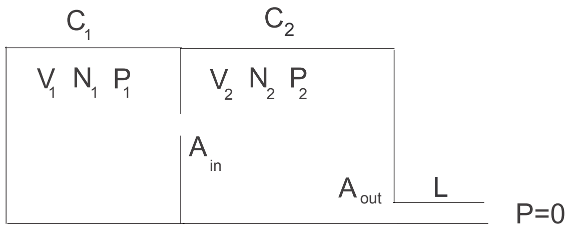


Figure 1: Sketch of the vacuum chamber system with associated model variables.

where N_2, F_{in}, F_{out}, and t are the number of He atoms in C_2, the particle flow from C_1 to C_2, the flow out of C_2 into the detector, and time, respectively.

The first question is now: how should we model the input particle flow values? A simple assumption would be to assume that the connection between C_1 and C_2 is through a small hole of area A_{in}. The second question is whether the transport is (a) fluid in nature (following a simple valve equation), or (b) following a kinetic-gas theory where the He atoms in C_1 obeys a perfect gas Maxwellian probability distribution. We will consider both cases but the conclusion for He-particle build-up in C_2 (and hence the whole dynamic process) will essentially be independent of this discussion, as we shall see.

Consider first case (a). The valve equation reads:

$$F_{in} \quad = cA_{in}\sqrt{P_1 - P_2},$$

where c is a constant. Since the time duration is rather small (and the hole area A_{in} is very small) it is reasonably to consider all thermodynamical parameters in C_1 independent of time t. Moreover, the pressure in C_2 is then always much smaller than the (constant) pressure in C_1. Hence,

$$F_{in} \quad \approx cA_{in}\sqrt{P_1} \approx constant.$$

Consider next case (b). A Maxwellian distribution of He atoms in C_1 corresponds to the probability distribution:

$$d\omega = \left(\frac{m}{2\pi k T_1}\right)^{3/2} \exp\left[-m\left(v_x^2+v_y^2+v_z^2\right)/(2kT_1)\right],$$

where m is the mass of a He atom, k is Boltzmann's constant, T_1 is the temperature in C_1, and v_x, v_y, v_z are the x, y, z particle velocities. This distribution is normalized, so that

$$\int d\omega dv_x dv_y dv_z = 1.$$

We then calculate the net number of He atoms escaping into C_2 through the hole per unit time ν as

$$\nu = \int_{-\infty}^{\infty} dv_x \int_{-\infty}^{\infty} dv_y \int_0^{\infty} dv_z v_z A_{in} \frac{N_1}{V_1} d\omega,$$

assuming the normal to the wall to be along the z direction and N_1, V_1 are the number of He atoms in C_1 and the volume of C_1, respectively. Notice that only particles with a positive z velocity will reach the wall per unit time, thus the integration over v_z starts from 0. The above integral can be easily carried out to give:

$$\nu = \frac{A_{in} P_1}{\sqrt{2\pi m k T_1}}.$$

Again, we have neglected the very small transport of He atoms back into C_1 from C_2 in the derivation of ν.

The important conclusion from the resulting expression in Equation (1) [case (b)] is that F_{in} is independent of time similar to the result found in case (a).

Let us next model the flow of He atoms out of C_2 moving towards the He detector. We shall do this assuming that the pipe connecting C_2 with the detector is a cylinder of length L and radius R. The dynamic viscosity of the He gas is denoted μ. Then, for a small radius and relatively slow rate of He leakage it is appropriate to consider transport laminar in nature such that Poiseuille flow conditions apply, i.e., the particle escape per unit time F_2 reads

$$\begin{aligned} F_2 &= \frac{\rho_2 \mathcal{N}_A}{4} \frac{\pi R^2}{32\mu L} \Delta P, \\ \Delta P &= P_2 - P_{detector} \approx P_2, \end{aligned}$$

where ρ_2, $\mathcal{N}_{\mathcal{A}}$ are the mass density of He atoms in C_2 and Avogadros' number, respectively, and $P_{detector} \approx 0$ due to vacuum conditions at the detector. The value of 4 in the denominator of the first factor stems from the fact that a He atom has a mass of 4 atomic mass units. Since the dynamic viscosity is usually constant with pressure at a given temperature while the kinematic viscosity $\eta = \mu/\rho$ is proportional to the inverse pressure ([2]), we find that F_2 is proportional to N_2^2 instead of N_2 (the latter would apply if the kinematic viscosity is independent of pressure)! This comes about as the perfect gas law states that particle number N_2 is proportional to pressure in C_2:

$$N_2 \quad = \tfrac{P_2 V_2}{kT_2},$$

where V_2 is the volume of C_2 since temperature T_2 in C_2 is assumed constant in time.

Hence, the expressions for F_1 and F_2 found above corresponds to the simple nonlinear differential equation:

$$\frac{dN}{dt} \quad = a - bN^2,$$

with a and b constants (easily found from the expressions above!). Supplementing Equation (1) with the initial condition:

$$N(t=0) \quad = 0,$$

immediately gives

$$N(t) \quad = \sqrt{\tfrac{a}{b}} \tfrac{\exp\left(2\sqrt{ab}t\right)-1}{\exp\left(2\sqrt{ab}t\right)+1},$$

In Figure 2, we plot the function $N(t)$ for different combinations of constants a and b.

Finally, the present model also allows an estimate of the *delay* in receiving a response at the He detector. Due to the separation L between C_2 and the He detector, a time delay τ exists equal to:

$$\tau \quad = \tfrac{L}{Q_{out}/A_{out}} = \tfrac{A_{out} 32 \mu L^2}{\pi R^2 P_2},$$

which – using the above conclusions on viscosity dependencies with pressure – shows that the time delay is inversely proportional to the particle number N_2 in C_2 and not a constant as assumed by Danfoss! However, if L is small

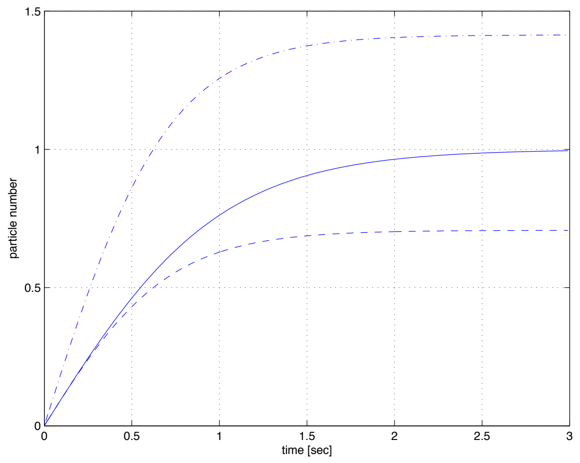


Figure 2: Time plots of the He particle number in C_2 (being a measure of the leakage) for three sets of parameters a and b computed using Equation (1). The solid, dash-dotted, and dashed curves correspond to: $a = 1, b = 1$; $a = 2, b = 1$; and $a = 1, b = 2$, respectively.

enough this delay time is close to zero. How can the observed time delay then be understood? A possible explanation is that the best model allows for a piping section connecting C_1 and C_2, i.e., a hole is not a sufficient description. Such a piping section will lead to a time delay which depends only on parameters in C_1 (again following a Poiseuille assumption). Since the C_1 parameters all are constants in time a constant time delay will result (in agreement with what is apparently observed!) Evidently, and intuitively trivial, such a piping section must have a length reflecting the geometry of the device in consideration and the leakage path, i.e., the time delay is strongly dependent on the actual device and the actual leakage path. An important additional conclusion is also that the parameter a depends on the leakage type (varying from product to product on the product line) while b depends on the measurement setup (geometry of the chamber C_2 and the pipe from C_2 to the He detector) but not the leakage type!

4 The $Q(t)$ Signal

We now summarize the model developed of the $Q(t)$ signal. During the time τ, when the signal travels to the detector, the measured Q-value is zero. Afterwards $Q(t)$ rises as in figure (2) towards an asymptotic value

Q_∞, with time dependence given by the function $N(t)$ of section 3.

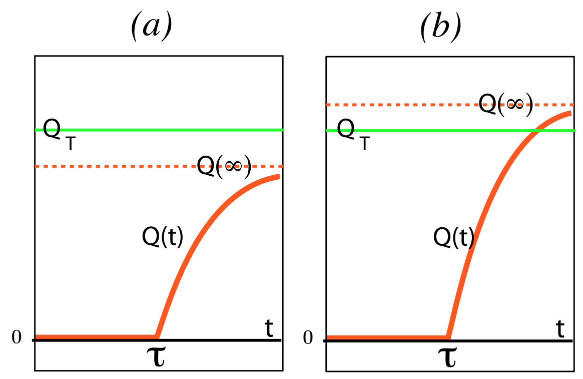


Figure 3: The ideal (noiseless) measurement in two situations: (a): The leak is small and $Q_\infty < Q_T$, (b): The leak is large and $Q_\infty > Q_T$.

Combining the pieces into a single expression, we have then:

$$Q(t) = Q_\infty(1 - \exp(-\delta(t - \tau))H(t - \tau), \tag{1}$$

where H is the Heaviside function (see, e.g.,[1]).

5 Leak tests and simulations

In what follows we will simulate data from Helium leak tests and compare various tests with respect to effectiveness and robustness. Since we have very limited knowledge of how e.g. Q_∞ vary in the production we are forced to make assumptions that is not supported by real world data and assign values to parameters in an almost arbitrary fashion. The result obtained from the simulations should therefore not be taken as quantitatively correct. What we will conclude nevertheless, is that there is a potential for solving Danfoss' problem by choosing a good test method.

We model the mesurement process as a process with stocastic deviations from the model developed in the previous section. Once we have a model of the process move on to simulating it and this allows to experiment with different tests and describe the test that performs best when fed with the simulated data.

5.1 Simulating the measurement process

We model the flow as a function of time is given by

$$Q(t) = Q_\infty(1 - \exp(-\delta(t-\tau))H(t-\tau), \tag{2}$$

where H is the Heaviside function.

When taking a mesurement it is most often the case that the measured value is not exactly equal to the physical value, but rather deviate by a small amount ϵ. In our case, we obtain the values $q_i = Q(t_i) + \epsilon_i$, for the flow measurements. The measurement error ϵ varies from mesurement to measurement and it is resonable to assume that it is a random variable, and that with a good aproximation it is normally distributed. Since there is no reason to expect that the measuring device be biased, we suppose the the mean value of ϵ is 0.

We assume that the standard deviation of ϵ is of the form $\sigma_\epsilon = c\sqrt{V}$ where V is the total amount that has flow through the device during the sample time δt. To see why this is reasonable suppose that the device was set to sample n times as often, and the flow is constant. Then we would obtain n mesurements $V/n + \epsilon_1, V/n + \epsilon_2, \ldots V/n + \epsilon_n$, where ϵ_i is the measurement error of these more frequent measurements. Adding the amounts up we get a total amount flowing through of $V + \epsilon_1 + \cdots + \epsilon_n$ in the time δt. Hence $\epsilon = \epsilon_1 + \cdots + \epsilon_n$. If we furthermore suppose that the ϵ_i are independent and normally distributed with mean 0 and standard variance σ', it follows that ϵ is normally distributed with the same mean, and with standard deviation $\sigma = \sqrt{n}\sigma'$. So the standard deviation of the error of the measurent of the amount V is $\sqrt{n}$ times the standard deviation of the amount V/n.

Due to presence of ambient Helium the mass spectrometer is calibrated to report zero flow at a certain level Q_{ref}. So when the device shows a level of $Q(t) + \epsilon$ the flow through the device is actually $Q_{\text{ref}} + Q(t)$. Therefore the standard deviation of the measurement error is $\sigma(Q(t), Q_{\text{ref}}) = c\sqrt{Q_{\text{ref}} + Q(t)}$.

Finally we assume that the different ϵ_i in one measurement series are independent. This assumption simplifies things greatly, but it might be wrong, especially if the hardware of the mass spectrometer does some filtering on the data.

We need to say something on how $Q_\infty, Q_{\text{ref}}, \delta$, and τ varies inside the population of measurements. In the absence of real world data, we make the simplest assumptions that agrees with the facts we do know. We assume that $\log(Q_\infty) \in \mathcal{N}(Q_\mu, Q_\sigma^2$, that is that $\log(Q)$ is normally distributed with mean μ_{Q_∞} and variance $\sigma_{Q_\infty^2}$. We suppose $\log(Q_{\mathrm{ref}}) \in \mathcal{N}(\mu_{Q_{\mathrm{ref}}}, \sigma^2_{Q_{\mathrm{ref}}})$. We also suppose $\tau \in \mathcal{N}(\mu_\tau, \sigma_\tau^2)$ and that δ is a constant. Finally we assume that all the mentioned random variables are independent.

To run a simulation, we need to assign values to the parameters. We have chosen the following

Parameter	Value
μ_{Q_∞}	$-\log(10^{-6})$
$\sigma_{Q_i nfty}$	0.28
$\mu_{Q_{\mathrm{ref}}}$	$-\log(10^{-6.5})$
$\sigma_{Q_{\mathrm{ref}}}$	0.25
μ_τ	2
σ_τ	0.25
δ	1
c	1.7×10^{-4}

The parameters are chosen so that approximately 0.4 percent of the products are defective.

5.2 The Maximum Test

The Maximum Test is the one currently employed by Danfoss. It considers the measurements $q_i = Q(t_i)$ obtained during the approximately 4 seconds the products is in the mass spectrometer. If one of these exceeds a treshold level Q_T the products is rejected. It means that the set where products are considered defective, the so called critical region, is of the form $\{(q_i) \mid \max_i(q_i) \geq Q_T\}$, and this is why the test is called the Maximum Test. If there were no variation is τ and δ and no measurement errors the test would work perfectly for a certain threshold Q_T^*, in the sense that all good products would pass the test and all defective products would be rejected.

However, when we incoporate measurement errors, taking the maximum gives higher values for noisy data than for exact data on the average, and therefore using the Maximum test with the treshold set to Q_T^* we expect that many good products are labelled as defective. This is confirmed by what we see when we simulate. When we use Q_T^* as treshold, around 8% of the good products are labeled defective. On the other hand, only very few defective products are labelled good (around 2 or 3 out $10,000$), so the test at this threshold is very conservative.

If we increase Q_T then the number good products that are rejected decreases, whereas the number of bad products that are accepted increases. To be able to compare the Maximum test to other test, we choose the value Q_T so that 1% of the defective products are accepted. With this choice 2.6% of the good products are flagged as defective which is a considerable improvement. In the next subsection we see we can improve even further.

5.3 The Average Test

We expect that there is considerable variance in the function $\max_i\{q_i\}$. This is because the maximum is realized by some q_{i^*} and changing it by a little is reflected by the same change in the maximum. On a small time interval, the function h does not vary much. If we take the average of the last values of the measured q_i we get a random variable with less variance (than the maximum) to base our decision on, and this is the idea of the Average test.

In the average test we take the average of the last measurements and accept the products if the value is below some threshold Q_{av} and otherwise reject it. Choosing the threshold so that 1% of the defective products are accepted, the Average test rejects approximately 1.4% of the good products.

5.4 The Regression Test

The Regression test works by considering the line segment that best approximates the last part of the data, in the sense of least squares. See Figure 4.

Calling the height of the rightmost point on the line segments for Q_R, the products is accepted if $Q_R < Q_T$ and otherwise rejected. If we calibrate the test so that 1% of the defective products pass the test, then around 1% of the good products are flagged as defective for the simulated data.

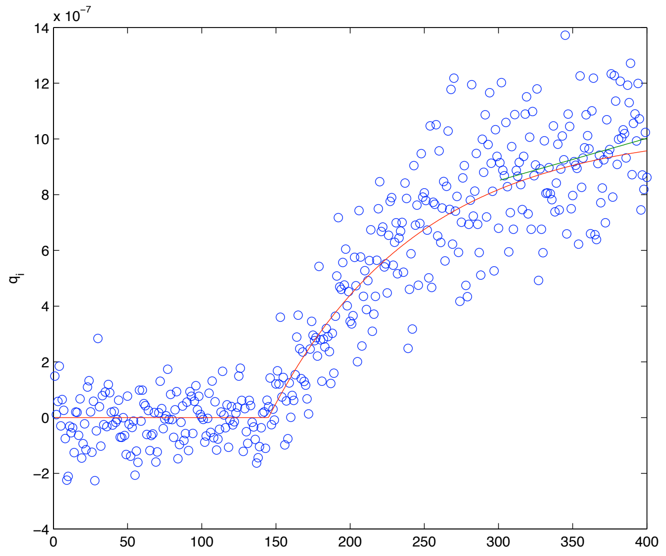


Figure 4: The curve shows an example of the flow for a product. The points around the curve forms the measured data; they are not lying on the curve because of measurement errors. The drawn line segment is the one that best approximates the last part of the measured data in the sense of least squares. The right end point of the curve is used by the Regresion test to decide whether to accept or reject the product.

5.5 The Frequency Test

The final test is an approximation of the best (or most powerful) test for the problem. Let $f(q_1, ..., q_n)$ denote the distribution of measured flows in a mesurement, $f(q_1, ..., q_n; 0)$ the distribution of measured flows in the population of good products and $f(q_1, ..., q_n; 1)$ the distribution of measured flows in the population of defective products. Define the region C_λ by

$$C_\lambda = \{(q_1, ..., q_n) \mid f(q_1, ..., q_n; 1) \geq f(q_1, ..., q_n; 2)\}.$$

We call the test that flag a product as defective if and only if the measure $(q_1, ..., q_n)$ lies in the critical set C_λ for the Frequency test. This test is optimal in the sense that any test that labels an equal percentage of good products as defective, will label a higher or equal percentage of defective products as good. This results follows from the Fundamental Lemma of Neyman and Pearson (see [3] chapter 3, page 166).

To implement the test, we need to calculate the two distributions. For compactness of notation, write $\mathbf{p} = (Q_\infty, Q_{\text{ref}}, \delta, \tau)$ for the parameters that vary between measurements of different products. Writing $g(\mathbf{p}, 0)$ for the distribution of the parameters for the population of good products and $g(\mathbf{p}, 1)$ for the distributions of parameters for the population of defective products. Let $\text{no}(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{x^2/2}$ denote the normal distribution and set

$$\phi_{\mathbf{p}}(q_1, ..., q_n) = \prod_i \text{no}\left(\frac{Q(t_i) - q_i}{\sigma_i}\right),$$

with $\sigma_i = c\sqrt{h(t_i)}$, and where $Q = Q_{\mathbf{p}}$ is the function defined by equation 2. Then $\phi_{\mathbf{p}}(q_1, ..., q_n)$ denote the conditional distribution of a measurement series $q_1, ..., q_n$ conditional to to the parameters are equal to $\mathbf{p}$. Therefore

$$f(q_1, ..., q_n; i) = \int_{\mathbf{p} \in \mathbb{R}^4} g(\mathbf{p}) \phi_{\mathbf{p}}(q_1, ..., q_n) d\mathbf{p},$$

$i = 0, 1$.

Since it seems difficult to evaluate the integral analytically, we proceed to find an approximation. Instead of integrating over Q_{ref} we set it to a typical value ${Q_{\text{ref}}}^* = \log(\mu_{Q_{\text{ref}}})$ and we suppose that δ is a constant (this is the case for the simulated data). The quadruble integral then reduces to a double integral,

$$f(q_1, ..., q_n; 0) \approx \int_{Q_\infty = -\infty}^{Q_A} \int_{\tau = -\infty}^{+\infty} g(Q, {Q_{\text{ref}}}^*, \tau, \delta) \phi_{\mathbf{p}}(q_1, ..., q_n) d\tau dQ_\infty,$$

and

$$f(q_1, ..., q_n; 1) \approx \int_{Q=Q_A}^{+\infty} \int_{\tau=-\infty}^{+\infty} g(Q, {Q_{\text{ref}}}^*, \tau, \delta) \phi_{\mathbf{p}}(q_1, ..., q_n) d\tau dQ_\infty.$$

Finally, we make variable substitutions to change the intervals of integrations to the unit interval and implement the test by carrying out the integrations numerically using Gauss-Legendre quadrature. Calibrating the test (which amounts to choosing the right value of λ) so that 1% of the defective products pass the test, we get that approximately 0.2% of the good products are labelled as defective.

5.6 Comparison of the Tests

We have run a simulation where (simulated) time series from $10,000$ good and $10,000$ defective products were fed to the tests. The following table summarises the test results.

	Max Test	Av. Test	Reg. Test	Fre. Test
Good rejected	266	164	108	19
Defective accepted	100	100	100	100
Rel. good rej.	2.7%	1.6%	1.1%	0.2%
Rel. defective rej.	1.0%	1.0%	1.0%	1.0%

We have also run a simulation where we choose ratio of defective products accepted to 0.02%. This very conservative ratio correspond to the the uncalibrated Maximum test. Because of the small amounts involved the numbers from the simulation should be regarded as quite rough approximations of the true expected values.

	Max Test	Av. Test	Reg. Test	Fre. Test
Good rejected	694	430	230	137
Defective accepted	2	2	2	2
Rel. good rej.	7%	4%	2%	1%
Rel. defective rej.	0.02%	0.02%	0.02%	0.02%

We see that with increasingly sophisticated methods we get increasingly better results. The three first methods have the advantage that they are easy to implement. Also, they assume little about how the time series should look if there were no noise, and they are in this sense robust. In the simulation we have assumed that τ and Q_∞ are independent.

A high value of Q_∞ and a high value of τ could give the same flow at the end of a measurement as a low value of Q_∞ coupled with a low value of τ. Since the three first methods all work by estimating the last point of the time series, they could reject a good product if τ is large and accept a bad product if τ is small. In the simulation τ and Q_∞ are independent. We have seen (in section 3) that we could expect τ and Q_∞ to be negatively correlated, and if this is the case we can expect the methods to work better.

We can qualitativly sum our findings up in the following table.

	Max Test	Av. Test	Reg. Test	Fre. Test
Power	*	**	**	***
Speed	***	**	**	*
Robustness	***	***	***	*

Here *power* refers to the efficiency of the test, i.e. a more powerful test rejects fewer good bottle than a less powerful test at a given level of accepting defective bottles. A test scores high in *speed* if it only requires few and simple calculations to carry out. A test is *robust* if it only makes few assumptions about the underlying probability distributions and is not very sensitive to changes in the parameters of these distributions.

At this point we don't have enough real world data to come with specific recommendation. But we can conclude on the basis of the simulation, that if we get information on the distributions of the parameters describing the products and the measurement errors, then a well chosen test on the basis of the time series of flow measurement can give significantly fewer good products labelled as defective, without letting more defective products pass the test.

6 Summary/ Conclusion

We have developed a three improved procedures for signal processing in a leak detection procedure.

Simulation results indicate that these new procedures, the number of false negatives will be significantly reduced, relative to the present situation. The price is introduction of a small number of false negatives. The more sophisticated (and therefore time consuming) the data processing is, the better the performance of the procedure.

——— o O o ———

A Matlab code

In the course of the work with the problem we did several simulations, and they were carried out in Matlab. The script `sim.m` is the main program. It calls the functions `GenGood.m`, `GenBad.m`, `Q_time.m`, and `Addnoise.m` to simulate time obtained by a mass spectrometer. Then it calls the functions `MaxTest.m`, `RegTest.m`, and `GaussTest.m` to simulate the different test methods.

A.1 sim

```
% This script simulates data from He leak tests
% and compares different measurement methods

close all
clear all
format compact;

% INITIALIZATION

Q_A = 2e-6; % The maximal permitted flow

N_good = 1e4; % no of 'good' products generated
N_bad = 1e4; % no of 'bad' products generated

% Variables describing the distribution of products

Q_mean = log(1e-6) % mean of log(Q)
Q_sigma = 0.28   % standard deviation of log(Q)
delta_mean  = 1
delta_sigma = 0
tau_mean = 2
tau_sigma = 0.25
Qref_mean = log(10^-6.5) % mean of log(Qref)
Qref_sigma = 0.25 % spread of log(Qref)
```

```
% Variables describing measurement of one product

% The standard deviation of the measurement error
% is determined by the constant c where
c = 1.7e-4;
measure_time = 4; % a measurement spans this many seconds
N_sample = 400; % no of flow measurements for one product

% Variables defining the various tests

Err2target = 0.01; % tests calibrated so they commit this ratio of error2

avsample = floor(N_sample/15);
% this many points are used for computing moving averages
% for the movavsample procedure

regsample = floor(N_sample/4);
% how many points are used for the regression test

% SIMULATION

% measurements are taken at times
T = measure_time/N_sample * (1:N_sample);
% We simulate time series
q_good = zeros(N_good, N_sample);
q_bad = zeros(N_bad, N_sample);
q_noisegood = zeros(N_good, N_sample);
q_noisebad = zeros(N_bad, N_sample);

fprintf('\nWe start generating data...\n');

[Qgood, taugood, deltagood, Qrefgood] = GenGood(...
    Q_mean, Q_sigma, tau_mean, tau_sigma, delta_mean, delta_sigma, ...
    Qref_mean, Qref_sigma, Q_A, N_good);
[Qbad, taubad, deltabad, Qrefbad] = GenBad(...
    Q_mean, Q_sigma, tau_mean, tau_sigma, delta_mean, delta_sigma, ...
    Qref_mean, Qref_sigma, Q_A, N_bad);

for i = 1:N_good
    q_good(i, :) = Q_time(Qgood(i), taugood(i), deltagood(i), T);
    q_noisegood(i, :) = Addnoise(q_good(i, :), Qrefgood(i), c);
end;

for i = 1:N_bad
    q_bad(i, :) = Q_time(Qbad(i), taubad(i), deltabad(i), T);
    q_noisebad(i, :) = Addnoise(q_bad(i, :), Qrefbad(i), c);
end;

% TESTS

% the Maximum test
display('Simulating the Maximum test');
[MaxError1, MaxError2] = MaxTest(q_noisegood, q_noisebad, Err2target);
% Error1 number of good products deemed bad
% Error2 number of bad products deemed good
MaxE1rel = 100*MaxError1 / N_good;
MaxE2rel = 100*MaxError2 / N_bad;

% the Average test
display('Simulating the Average test');
[AError1, AError2] = ...
    AvrTest(q_noisegood, q_noisebad, avsample, Err2target);
% MAError1 number of good products deemed bad
% MAError2 number of bad products deemed good
AE1rel = 100*AError1 / N_good;
AE2rel = 100*AError2 / N_bad;

% the Regression test
display('Simulating the Regression test');
[RegError1, RegError2] = ...
    RegTest(q_noisegood, q_noisebad, regsample, Err2target);
% RegError1 number of good products deemed bad
% RegError2 number of bad products deemed good
RegE1rel = 100*RegError1 / N_good;
RegE2rel = 100*RegError2 / N_bad;

% the Frequency test / Gauss test
display('Simulating the Frequency test');
[FreError1, FreError2] = GaussTest(q_noisegood, q_noisebad, ...
    Q_mean, Q_sigma, Q_A, tau_mean, tau_sigma, ...
    Qref_mean, Qref_sigma, delta_mean, c, T, Err2target);
% Error1 number of good products deemed bad
% Error2 number of bad products deemed good
FreE1rel = 100*FreError1 / N_good;
FreE2rel = 100*FreError2 / N_bad;

% OUTPUT

% finally we out put the results

fprintf('\nThe generated data is formed by %i good products', N_good);
fprintf(' and %i bad products.\n', N_bad);
fprintf('\nResults of tests\n');
display('               Max test  Avr Test   Reg Test   Fre Test');
fprintf('Good rejected    %5i     %5i     %5i     %5i\n', ...
    MaxError1, AError1, RegError1, FreError1);
fprintf('Bad accepted     %5i     %5i     %5i     %5i\n', ...
    MaxError2, AError2, RegError2, FreError2);
fprintf('Good rej %%       %5.2f%%    %5.2f%%    %5.2f%%    %5.2f%%\n', ...
    MaxE1rel, AE1rel, RegE1rel, FreE1rel);
fprintf('Bad accpt %%      %5.2f%%    %5.2f%%    %5.2f%%    %5.2f%%', ...
    MaxE2rel, AE2rel, RegE2rel, FreE2rel);
fprintf('\n\n');
```

A.2 GenGood

```
function [Q, tau, delta, Qref] = GenGood(...
    Q_mean, Q_sigma, tau_mean, tau_sigma, delta_mean, delta_sigma, ...
    Qref_mean, Qref_sigma, Q_A, F_sample)
% computes Q, tau, delta, Qref vectors
% Q, tau, delta, Qref are independent
% Q is picked  so that log(Q) is normal distr. with mean Q_mean
% and spread Q_sigma. Q is picked from the population with Q < Q_A
% tau is normally distributed according to tau_mean and tau_sigma
% delta is normally distributed according to delta_mean and delta_sigma
% log(Qref) is normally distr. accroding to Qref_mean and Qref_sigma
% F_sample is the number of samples generated.

delta  = randn(F_sample, 1)*delta_sigma + delta_mean;
```

```
tau = randn(F_sample, 1)*tau_sigma + tau_mean;
Qref = exp(randn(F_sample, 1)*Qref_sigma+Qref_mean);

i = 1;
while i <= F_sample
    Q(i, 1) = exp(randn()*Q_sigma + Q_mean);
    if Q(i, 1) < Q_A
        i = i + 1;
    end;
end;
```

A.3 GenBad

```
function [Q, tau, delta, Qref] = GenBad(...
    Q_mean, Q_sigma, tau_mean, tau_sigma, delta_mean, delta_sigma, ...
    Qref_mean, Qref_sigma, Q_A, F_sample)
% computes Q, tau, delta, Qref vectors
% Q, tau, delta, Qref are independent
% Q is picked  so that log(Q) is normal distr. with mean Q_mean
% and spread Q_sigma. Q is picked from the population with Q > Q_A
% tau is normally distributed according to tau_mean and tau_sigma
% delta is normally distributed according to delta_mean and delta_sigma
% log(Qref) is normally distr. accroding to Qref_mean and Qref_sigma
% F_sample is the number of samples generated.

delta  = randn(F_sample, 1)*delta_sigma + delta_mean;
tau = randn(F_sample, 1)*tau_sigma + tau_mean;
Qref = exp(randn(F_sample, 1)*Qref_sigma+Qref_mean);

S = rand(F_sample, 1);
P = zeros(F_sample, 1);
P_cut = (log(Q_A) - Q_mean) / Q_sigma;
opts = optimset('display', 'off');
for i = 1 : F_sample
    F = @(x) ...
        (erf(x/sqrt(2))-erf(P_cut/sqrt(2)))/(1 - erf(P_cut/sqrt(2))) ...
        - S(i);
    P(i ,1) = fsolve(F, P_cut, opts);
end;
```

```
Q = exp(P*Q_sigma + Q_mean);
```

A.4 Q_time

```
function q = Q_time(Q, tau, delta, T)
% Generate a time series
% q(t) = Q(1-exp(-delta*(t-tau))*H(t - tau), for t in T.
q = Q*(1 - exp(-delta*(T-tau))).*(1 + sign(T - tau)) / 2;
```

A.5 Addnoise

```
function qnoise = Addnoise(q, Qref, c)
% Adds a normally distributed random variable of expected value 0
% and standard deviation c*sqrt(q(i) + Qref) to each element q(i)
% of q.
sigma = c * sqrt(q + Qref);
noise = sigma.*randn(1, length(q));
qnoise = q + noise;
```

A.6 MaxTest

```
function [e1, e2] = MaxTest(Qgood, Qbad, Err2target)

[Nbad, N] = size(Qbad);
[Ngood, N] = size(Qgood);

% First we calibrate the test so that
% the fraction Errr2target of the bad are tested 'good'
M = max(Qbad');
M = sort(M);
i = floor(Err2target * Nbad);
if i < 1
    i = 1;
end;
Q_T = 0.5 * (M(i) + M(i+1)); % the treshold

% Then we count the number of errors commited
```

```
Error1 = 0; % Error1 fraction of good products deemed bad
Error2 = 0; % Error2 fraction of bad products deemed good

M = max(Qgood’);
for i = 1:Ngood
    if M(i) >= Q_T
        Error1 = Error1 + 1;
    end;
end;

M = max(Qbad’);
for i = 1:Nbad
    if M(i) < Q_T
        Error2 = Error2 + 1;
    end;
end;

e1 = Error1;
e2 = Error2;
```

A.7 RegTest

```
function [e1, e2] = RegTest(qgood, qbad, regsample, Err2target)
% The regression test. Uses linear regresion on the last avsample points
% to determine to accept or reject
[Nbad, N] = size(qbad);
[Ngood, N] = size(qgood);

% First we calibrate the test so that
% the fraction Errr2target of the bad are tested ’good’
A = ones(regsample, 2);
for i = 1 : regsample
    A(i, 2) = i;
end;
w = [1 regsample] * (A’ * A)^(-1) * A’;
Qestbad = qbad(:, N - regsample +1 : N) * w’;
Qestbad = sort(Qestbad);
i = floor(Err2target * Nbad);
if i < 1
```

```
    i = 1;
end;
Q_T = 0.5 * (Qestbad(i) + Qestbad(i+1)); % the treshold

% Then we count the number of errors commited

Error1 = 0; % Error1 fraction of good products deemed bad
Error2 = 0; % Error2 fraction of bad products deemed good

Qestgood = qgood(:, N - regsample +1 : N) * w';
for i = 1:Ngood
    if Qestgood(i) >= Q_T
        Error1 = Error1 + 1;
    end;
end;

for i = 1:Nbad
    if Qestbad(i) < Q_T
        Error2 = Error2 + 1;
    end;
end;

e1 = Error1;
e2 = Error2;
```

A.8 GaussTest

```
function [e1, e2] = GaussTest(qgood, qbad, Q_mean, Q_sigma, Q_A, ...
    tau_mean, tau_sigma, Qref_mean, Qref_sigma, delta, c, T, Err2target)
% Simplified frequency test.
% A likelihood ratio test that uses Gauss-Legendre quadrature
% to aproxmiate the ratio.

% We find the number of 'bad' samples
[Nbad, N] = size(qbad);
% and good samples
[Ngood, N] = size(qgood);
% and the typical deviation of measurement error along a timeseries of
% flow rates
sigma_typ = c*sqrt(exp(Q_mean) + ...
    Q_time(exp(Q_mean), tau_mean, delta, T))';

% Then we set up the Gauss-Legendre quadrature of order
M = 21;
% for integrating between 0 and 1. The abscissas are given by
x = [      0.0031239146898052498698789820310295354034; ...
           0.0163865807168468528416888925461524192876; ...
           0.0399503329247995856049064331425155529204; ...
           0.0733183177083413581763746807062161648620; ...
           0.1157800182621610456920610743468859825895; ...
           0.1664305979012938403470166650048304187015; ...
           0.2241905820563900964704906016378433566890; ...
           0.2878289398962806082131655557281059739518; ...
           0.3559893415987994516996037419676998400455; ...
           0.4272190729195524545314845088306568349418; ...
           0.5000000000000000000000000000000000000000; ...
           0.5727809270804475454685154911693431650582; ...
           0.6440106584012005483003962580323001599545; ...
           0.7121710601037193917868344442718940260482; ...
           0.7758094179436099035295093983621566433110; ...
           0.8335694020987061596529833349951695812985; ...
           0.8842199817378389543079389256531140174105; ...
           0.9266816822916586418236253192937838351380; ...
           0.9600496670752004143950935668574844470796; ...
           0.9836134192831531471583111074538475807124; ...
           0.9968760853101947501301210179689704645966];
% and the weights are
w = [     0.008008614128887166662112308429235508133924; ...
          0.01847689488542624689997533414966483635729; ...
          0.02856721271342860414181791323622398097488; ...
          0.03805005681418965100852582665009158023242; ...
          0.04672221172801693077664487055696604109659; ...
          0.05439864958357418883173728903505282170669; ...
          0.06091570802686426709768358856286679903582; ...
          0.06613446931666873089052628724838780222205; ...
          0.06994369739553657736106671193379155545011; ...
          0.07226220199498502953191358327687627180509; ...
          0.07304056682484521359599257384168559412240; ...
          0.07226220199498502953191358327687627180509; ...
          0.06994369739553657736106671193379155545011; ...
          0.06613446931666873089052628724838780222205; ...
          0.06091570802686426709768358856286679903582; ...
          0.05439864958357418883173728903505282170669; ...
          0.04672221172801693077664487055696604109659; ...
          0.03805005681418965100852582665009158023242; ...
          0.02856721271342860414181791323622398097488; ...
          0.01847689488542624689997533414966483635729; ...
          0.008008614128887166662112308429235508133924];


% the abscissas for the tau integration
opts = optimset('display', 'off');
for i = 1 : M
    er = @(t) 1/2*(1 + erf(t/sqrt(2))) - x(i);
    s(i) = fsolve(er, 0, opts);
end;
Tau = tau_mean + tau_sigma * s;

% The relative number of 'good' products is
gam = 1/2 * (1 + erf((log (Q_A) - Q_mean) / (sqrt(2)*Q_sigma)));
% we compute the abscissas, Q2, for the good likelihood
for i = 1 : M
    er = @(t) 1/2*(1 + erf(t/sqrt(2))) - gam*x(i);
    s(i) = fsolve(er, 0, opts);
end;
Q2 = exp(Q_mean + Q_sigma * s);
% and for the bad likelihood they are
for i = 1 : M
    er =  @(t) 1/2*(1 + erf(t/sqrt(2))) - (gam + (1-gam)*x(i));
    s(i) = fsolve(er, 0, opts);
end;
Q1 = exp(Q_mean + Q_sigma*s);
% At the abscissas the time series for Q are q1t and q2t:
for i = 1 : M
    for j = 1 : M
        q1t(:, i, j) = Q_time(Q1(i), Tau(j), delta, T);
        sigma1t(:, i, j) = c * sqrt(q1t(:, i, j) + exp(Qref_mean));
        q2t(:, i, j) = Q_time(Q2(i), Tau(j), delta, T);
        sigma2t(:, i, j) = c * sqrt(q2t(:, i, j) + exp(Qref_mean));
    end;
end;

% We calculate the likelihood ratio for the 'bad' data to calibrate
% the method

for k = 1 : Nbad
    T = 0; N = 0;
    for i = 1 : M
        for j = 1 : M
            T = T + w(i)*w(j)* ...
              prod(sigma_typ./sigma1t(:,i,j)) * ...
              exp(-sum((qbad(k, :)'-q1t(:, i, j)).^2 ./ ...
              (2*sigma1t(:,i,j).^2)));
            N = N + w(i)*w(j)* ...
              prod(sigma_typ./sigma2t(:,i,j)) * ...
              exp(-sum((qbad(k, :)'-q2t(:, i, j)).^2 ./ ...
              (2*sigma2t(:,i,j).^2)));
        end;
    end;
    % log of the likihood ratio is Alpha where
    if N > 0
        Alpha_bad(k) = log(T) - log(N);
    else
        Alpha_bad(k) = 1e3; % +infinity
    end;
end;
Alpha_sort = sort(Alpha_bad);
i = floor(Err2target * Nbad);
if i < 1
    i = 1;
end;
Alpha_T = 0.5 * (Alpha_sort(i) + Alpha_sort(i+1)); % the treshold

% We calculate log of the likelihood ratio for the 'good' data
for k = 1 : Ngood
    T = 0; N = 0;
    for i = 1 : M
        for j = 1 : M
```

```
            T = T + w(i)*w(j)* ...
              prod(sigma_typ./sigma1t(:,i,j)) * ...
              exp(-sum((qgood(k, :)'-q1t(:, i, j)).^2 ./ ...
              (2*sigma1t(:,i,j).^2)));
            N = N + w(i)*w(j)* ...
              prod(sigma_typ./sigma2t(:,i,j)) * ...
              exp(-sum((qgood(k, :)'-q2t(:, i, j)).^2 ./ ...
              (2*sigma2t(:,i,j).^2)));
        end;
    end;
    if T > 0
        Alpha_good(k) = log(T) - log(N);
    else
        Alpha_good(k) = -1e3; % - infinity
    end;
end;

Error1 = 0; % Error1 fraction of good products deemed bad
Error2 = 0; % Error2 fraction of bad products deemed good

for i = 1 : Ngood
    if Alpha_good(i) >= Alpha_T
        Error1 = Error1 + 1;
    end;
end;

for i = 1:Nbad
    if Alpha_bad(i) < Alpha_T
        Error2 = Error2 + 1;
    end;
end;

e1 = Error1;
e2 = Error2;
```

References

[1] A.C. Fowler: *Mathematical Models in the Applied Sciences*, Cambridge Texts in Applied Mathematics (1995).

[2] Landau and Lifshiz: *Fluid Mechanics*, 2. ed., Pergamon Press (1993).

[3] L. Schmetterer: *Introduction to Mathematical Statistics,* Springer Verlag, Berlin Heidelberg (1974).