Informe final del proyecto HC-1204.1

Evaluación de dos métodos emergentes para monitorear el nivel del agua

Serge Tamari

5 de marzo, 2013

SECRETARÍA DE MEDIO AMBIENTE Y RECURSOS NATURALES

SEMARNAT

Parte 1. Artículo sobre el monitoreo en lagos

Parte 2. Artículo sobre el monitoreo en tinacos

Parte 3. Código para adquirir datos en lagos

Parte 4. Código para adquirir datos en tinacos

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Testing a near-infrared Lidar mounted with a large incidence angle to monitor the water level of turbid reservoirs

S. Tamari[a,*], J. Mory[b], V. Guerrero-Meza[c]

[a] Instituto Mexicano de Tecnología del Agua, Paseo Cuauhnáhuac 8532, Jiutepec Mor. 62550, Mexico
[b] Ecole des Mines de Douai, 941 rue Charles Bourseul, 59508 Douai, France
[c] Disime S.A. de C.V., Plaza Guardiola 94, Col. Alfonso Ortiz Tirado, México DF 09020, Mexico

ARTICLE INFO

Article history:
Available online 15 March 2011

Keywords:
Terrestrial Lidar
Off-Nadir angle
Tyndall effect
Stage monitoring
Lake
Earth–dam embankment

ABSTRACT

It is sometimes difficult to monitor the water level of reservoirs with a sloping bank – such as many lakes and earth-dam embankment – using terrestrial techniques or airborne instruments. A proposed alternative is a new technique using a terrestrial near-infrared Lidar mounted with a large incidence angle (at least between 40° and 70°). This technique assumes that the Lidar can detect the (sub-)surface of a water body provided that it contains enough suspended particles to backscatter the light emitted by the instrument to its detector. Tests performed with a commercial Lidar show that the technique can be used to estimate the water level of a reservoir with moderate accuracy (within ±0.05 m [$p = 0.95$]) when the water is very turbid (Secchi depth < 0.5 m). The versatility and accuracy of the technique is expected to improve in the future with the use of current Lidar that are more sophisticated than the tested one.

1. Introduction

Classical techniques to monitor the stage of water bodies are based on terrestrial instruments (IOC, 2006; ISO, 2008; Vuglinskiy, 2009). Most of these techniques use an emerged transducer that must be mounted vertically (such as systems with an ultrasonic transducer, a radar, or even a Lidar). The transducer determines the distance to the water surface by sending a signal that is echoed by water, as a mirror would; however, this kind of mounting is not practical for water bodies with a sloping bank, which is the case of many lakes and earth-dam embankments. In said case, certain types of measuring systems that are partly submerged in water could be used (such as systems with a submerged pressure transducer, a bubbling pipe connected to an emerged pressure transducer, or even an upward looking ultrasonic transducer); however, the submerged parts can be damaged by water due to clogging, corrosion and/or incrustation. In addition, it is sometimes difficult to protect the emerged parts (such as cables or tubing) from bad weather and vandalism.

Modern techniques using airplanes or satellites are now attractive alternatives to monitor the stage of water bodies with a sloping bank. A particular technique consists in floating a buoy with a GPS on the water surface (IOC, 2006); however, said technique continues to be costly and a buoy floating into a lake or a reservoir is exposed to vandalism. Other techniques using airborne instruments (such as radar altimeter and Lidar) have been used successfully to monitor the stage of some water bodies (Huang, 1981; Alsdorf et al., 2007; Höfle et al., 2009; Vuglinskiy, 2009); however, they also have some disadvantages, including: the time interval between measurements taken at the same site is sometimes long (several days), small water bodies are often missed, the measurements can be influenced by weather conditions, and field data are sometimes needed to calibrate airborne instruments.

For the above reasons, it would be convenient to be able to monitor the stage of water bodies with a sloping bank using an instrument mounted at the edge. In this context, a side-looking radar could be used (Fulton and Ostrowski, 2008); however, such an instrument does not detect the smooth surface of still water-bodies, and is therefore difficult to use in lakes and reservoirs. Another solution would be to use a digital camera to detect certain contrasts between the water and the ground (by visible and/or by thermal-infrared imagery); however, data processing is not straightforward (Matthies et al., 2003). Passive optical measurements have also been proposed to monitor the stage of water bodies (Bills et al., 2007); however, this technique requires determining the bottom reflective properties of the considered water body. Thus, this study reports on a new technique to monitor the stage of water bodies with a sloping bank; it requires the use of a terrestrial near-infrared Lidar and only works for turbid waters.

* Corresponding author. Tel.: +52 777 329 36 00.
E-mail addresses: tamari@tlaloc.imta.mx (S. Tamari), julien.mory@minesdedouai.fr (J. Mory), vguerrero@disime.com.mx (V. Guerrero-Meza).

2. Background

2.1. Lidar operating principle to measure distance

A Lidar (*LIght Detection And Ranging*) is an active remote sensing system that uses a laser to send a pulse of light of a given wavelength – usually between 300 nm (near-ultraviolet) and 1400 nm (near-infrared) – to a reflective target, and then measures some properties of the light sent back by the target (such as the time shift and the Doppler shift). A Lidar can determine the distance, speed, and/or composition of several kinds of objects. These instruments have been used in different ways (including from a fixed point on the land, a terrestrial vehicle, an airplane, a satellite, or underwater). They can be more or less sophisticated (such as having one or more lasers, one or more light-detectors, and more or less ability to process over time the light sent back by a target). Among other applications, the Lidar systems have been extensively used to monitor the stage of water bodies (Guenther et al., 2001; Alsdorf et al., 2007; Höfle et al., 2009).

This study considers the simplest Lidar configuration. The instrument is set at a fixed position in the air and measures the time it takes for a pulse of light to travel from the instrument to a target and back again (Δt, s). The time-of-flight (*TOF*) method is used to compute the distance between the Lidar and the target by assuming that the light travels straight into the air and by knowing the speed of the light through the air (c, m/s); said distance (L, m) is $L = c \times \Delta t/2$.

2.2. Light-reflection by the surface of natural water bodies

Two extreme cases of light-reflection by a surface must be considered: specular (in which the surface reflects the light as a mirror) or diffuse (where the surface reflects the light in any direction). In the first case, a Lidar should be mounted with a zero incidence angle to detect the surface of a target, while in the second case, the instrument could detect the surface even if it is mounted with a non-null incidence angle. In reality, the light reflection by natural surfaces is between specular and diffuse, so the practical question here is to know whether or not the surface of natural water-bodies could behave as a diffuse surface with respect to the light sent by a Lidar.

Unfortunately, the surface of natural water bodies is classically considered as specular, unless there is floating matter (such as sea whitecaps, fresh ice, foam or oil). Therefore, airborne Lidar systems (*ALS*) are commonly mounted with a near-zero incidence angle ($\theta < 10°$); such an application is well documented (Alsdorf et al., 2007; Höfle et al., 2009; Allouis et al., 2010). For this reason also, the terrestrial Lidar-systems mounted with a large incidence-angle ($\theta > 30°$) hardly detect the water bodies (Matthies et al., 2003; Heritage and Hetherington, 2007).

It as been recognized that the surface of an agitated water body behaves to some extent as if it were diffuse with respect to the light sent by a Lidar (Menzies et al., 1998; Guenther et al., 2001; Carter et al., 2001; Höfle et al., 2009; Li et al., 2010). This is because the water surface in this case is no longer smooth; strictly speaking, this is not a truly diffuse behavior, but a particular case of specular reflection in which the facets of the water surface that are perpendicular to the Lidar reflect a part of the light back to the instrument (Fig. 1a). This situation – known as the *Bragg scattering* – may occur in the sea (due to the wind-induced waves) or in rivers (due to turbulence). In said cases, airborne Lidar systems can be mounted with a non-null but small incidence-angle ($\theta < 25°$). Nonetheless, such a situation is not considered relevant for still water-bodies such as lakes and dam embankments (Carter et al., 2001; Höfle et al., 2009).

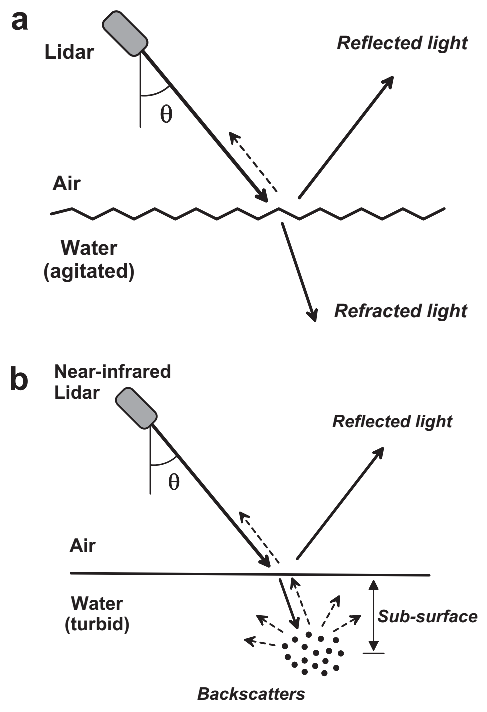

Fig. 1. Conceptual sketches of (sub-)surface water detection by a Lidar with a non-null incidence angle (θ): (a) commonly considered case, i.e. a non-smooth water surface (due to waves and/or turbulence) that reflects a part of the Lidar signal back to the instrument; (b) case considered in this study, i.e. a smooth water-surface of turbid water, into which the near-infrared Lidar signal is refracted and backscattered by the suspended particles that are just below the water surface.

2.3. Others interactions between light and water bodies

A portion of the light sent by a Lidar is not reflected by the surface of a water body, but rather penetrates into the water by refraction. In this case, the angle of penetration of light is reduced (according to Snell's law) and the light intensity decreases as the depth of penetration increases (according to Beer–Lambert's law).

Light absorption by water depends on light color. In the range of colors emitted by Lidar, blue-green ($\lambda \approx 540$ nm) is less absorbed by water. Thus, many airborne Lidar systems use an ultraviolet or a green laser to detect the bottom of shallow water bodies (to a depth of several meters). At the opposite extreme, the near-infrared ($\lambda \approx 950$ nm) is strongly absorbed by water (e.g., Matthies et al., 2003; Bills et al., 2007). In this case, the literature suggests that the current Lidar cannot detect a near-infrared signal that has penetrated through a layer of clean water at a depth between ≈0.2 m and 0.5 m (Matthies et al., 2003; Höfle et al., 2009; Allouis et al., 2010); the depth of said layer is expected to be smaller when the water is turbid (Guenther et al., 2001; Matthies et al., 2003; Alsdorf et al., 2007; Höfle et al., 2009).

While the light is not completely absorbed by the water, it may be scattered in several ways by the water itself (according to the scattering mechanisms described by Rayleigh, Ramann and Brillouin). Some Lidar can detect this scattered light – whose frequency may be different than that of the incident light – to analyze some water properties. The light may also be scattered in different directions by small particles suspended in the water, such as colloidal mineral particles (Doxaran et al., 2002), plankton (Churnside and Donaghay, 2009), and air bubbles (Zhang et al., 1998; Churnside, 2010). Such a phenomenon, known as the *Tyndall effect*, is partly explained by the Mie scattering-theory (which considers spherical particles). Several studies that have been performed with airborne

Lidar systems are related to this effect, e.g., it makes it possible to characterize sea clarity (Phillips and Koerber, 1984) and detect turbid zones in the sea (Churnside and Donaghay, 2009); such an effect is also problematic when trying to detect an object with an under-water Lidar imaging system (Caimi and Dalgleish, 2010).

However, only a few authors (Menzies et al., 1998; Li et al., 2010) have studied the consequences of the light backscattered by suspended particles on the signal received by a Lidar mounted with a large incidence angle ($\theta > 30°$). In particular, it was demonstrated recently that such a "water subsurface contribution" should be the main cause of signal return to an ultraviolet airborne Lidar mounted over the sea with an incidence angle larger than 30° (Li et al., 2010). It is worth noting that the intensity of the light backscattered by turbid water depends on light color and water composition. For instance, coastal waters with mineral suspended-particles were found to be more reflective to the green than to the near-infrared when the particle concentration is low, and more reflective to the near-infrared than to the green when the particle concentration is high (Doxaran et al., 2002).

For the above reasons, it was thought that due to the scattering produced by small particles suspended in water, the light sent by a near-infrared Lidar mounted with a large incidence angle could penetrate slightly below the surface of a turbid water body and then return to the instruments (Fig. 1b). If so, it would be possible to use such an instrument to monitor the (sub-)surface of turbid water bodies with a sloping bank. As far as we know, a few authors have considered such an application (Heritage and Hetherington, 2007), but its feasibility still has not been demonstrated. Some results obtained regarding this application are shown and discussed below.

3. Material and methods

3.1. The tested Lidar

The ability of two commercial Lidar to monitor the stage of turbid water was tested: a *Watchman* model (version 3100-SR, firmware 35-AWLX-2.0, Optech Inc., Vaughan, Canada) and a *Sentry* model (version SR, firmware SR-V2.8, Optech Inc.).[1] The Sentry Lidar was often unable to detect the water surface or gave erratic results. We believe that this is related to the internal configuration of the instrument (its detector-sensitivity and firmware), which is not available to users. Therefore, only the results obtained with the Watchman Lidar are shown below.

According to its user's manual (Optech, 2004), the Watchman Lidar uses a near-infrared laser ($\lambda = 905$ nm, beam divergence = 0.28°, Class 1M) to determine distances from a target within a range between ≈0.2 and 250 m. Measurement uncertainty is stated as ±0.04 m [$p = 0.95$] for a measuring rate of one reading per second. The user's manual describes the Lidar as "an object positioner designed for industrial use" and specifies that its laser beam can be reflected from a diffuse object "at virtually any angle and still return to the unit to produce a range measurement"; however, no specific comment is made about the ability of the instrument to detect a water surface.

When being used, the tested Lidar was powered continuously with a 12-V battery (model PS-100, Campbell Scientific, Logan, USA). After each reading, the Lidar returned the raw measured distance (L, m) and two diagnostic-parameters: the dropout number (which indicates the number of laser shots that do not return to the instrument during a reading) and the signal intensity (which is, an index of the light intensity received by the Lidar detector). All these data were stored using a datalogger (model CR-800, Campbell Scientific) connected to the serial port of the Lidar.

A few options are available for the user to configure the tested Lidar (Optech, 2004). After some preliminary tests, the instrument was set with the following configuration: one reading per second (a measuring rate low enough to determine distances with the best stated uncertainty), "direct response" (successively acquired data were not averaged together) and "filtered last-pulse" (recommended by the manufacturer when the Lidar is used in an environment with "thick rain, steam, fog or dust"). It is worth noting that the user's manual warns that the Lidar may be less accurate when it is set with the "filtered-last-pulse" option, but it is also suggested that this would happen with a high "dropout number".

3.2. Laboratory tests

Before testing the ability of the Watchman Lidar to detect a water surface, some preliminary laboratory tests were performed with this instrument. First, the ability of the Lidar to detect a solid target placed at a distance of several meters was checked. On one hand, the instrument could not always detect an highly reflective surface (such as an aluminum plate). In said case, a "saturation" error message was sent indicating that the return signal intensity was too high. On the other hand, the instrument had no difficulty detecting a moderately reflective surface (such as a sheet of white paper). In this case (Tamari et al., 2010), the Lidar was able to measure distances up to 50 m within ±0.02 m. When a paper sheet was set for a few days in front of the Lidar at a distance of ≈1.5 m, sudden changes in the measured distance were observed. These changes had a magnitude of about ≈0.01 m and a periodicity of ≈12 h, and were interpreted as electronic instabilities in the instrument. However, all the above results were consistent with the technical specifications of the tested Lidar (Optech, 2004).

After the preliminary tests, others were performed on a water tank (≈2.6 m deep, 4.5 m long and 0.9 m wide) to investigate the ability of the Lidar to detect the surface of a water body. First, the Lidar was checked for its ability to measure the distance to the surface of clean tap water, with an uncertainty of less than ±0.04 m [$p = 0.95$] when the instrument was mounted vertically above the water surface; as expected, this did not hold true when it was inclined (see Section 2.2). A series of tests were then performed using turbid water and the Lidar mounted with a large incidence angle.

To prepare turbid water, the tank was filled with tap water, into which solid particles were poured and mixed. The solid particles were cement plaster or a natural silty material (which is extracted from a volcanic-ash soil and which is known as *tepetate* in Mexico). Before starting a test, special care was taken to remove all the floating matter (such as light flocculated aggregates and organic matter). The tests then consisted of successively draining and filling the tank from the bottom several times. At the end of each filling, a portion of the water was evacuated through a weir located at the upper part of the tank; the amount of suspended particles was therefore decreased, lowering the water turbidity. The stage varied between ≈0.4 m (to avoid possible interferences between the Lidar signal and the tank bottom) and <2.5 m (to ensure that the minimum distance of the water surface with respect to the Lidar was greater than ≈0.5 m).

The duration of tests performed with the Lidar mounted on the water-tank with a large incidence angle was between 60 and 180 mins. A times, the tank draining or filling was stopped in order to manually check the stage (using a graduated scale) and to estimate water clarity (see Section 3.4). A manometric pressure transducer (model DCX-16VG with a range of 50 kPa, Keller, Newports News, USA) was used as a reference to monitor the stage (every 1 s). Said instrument can estimate the stage of a clear-water tank

[1] Any reference to a commercial model and its manufacturer is for product identification purposes only, and does not represent an endorsement of the product.

with an uncertainty better than ±5 mm [$p = 0.95$] (Tamari and Aguilar-Chávez, 2010). Because no correction was made for the increase in water density due to the presence of suspended particles, the pressure transducer was theoretically overestimating the stage during the tests performed with turbid water; however, such an overestimation was less than 5 mm (considering that the amount of suspended particles into the tank was less than 10 kg). During the tests, the distance measured by the Lidar was recorded every 2 s. Tests were performed with the Lidar mounted with an incidence angle of $\approx 50°$, 60° and 70°.

3.3. Field experiment

An experiment was performed for nine days (September 19–27, 2010) to investigate the ability of the Watchman Lidar to detect the surface of a water body under field conditions (Fig. 2). The Tuxpan dam-embankment site (19°32′26.10″N–100°29′9.30″W) was selected for several reasons: there is a vertical wall onto which a Lidar can be easily mounted with a small or a large incidence angle, water is visually turbid, the stage is routinely monitored using a graduated scale (every hour) and an ultrasonic transducer (every 15 min), and the embankment is small (with a capacity of $\approx 5 \times 10^6$ m^3) so that the stage may vary quickly when it is raining.

During the experiment, an automatic rain gauge (with a sensitivity 0.25 mm, High Sierra Electronics, Grass Valley, USA) was installed at the monitoring site to determine whether the Lidar measurements were more erratic when it rains (as suggested by the Lidar user's manual, and by the literature: e.g., Guenther et al., 2001).

An ultrasonic water-level transducer (model Vantage 2200/FB5 with a range of 8 m, Eastech Flow Controls, Upper Saddle River, USA) was used as a reference to monitor every 15 min. Said instrument had an uncertainty better than ±20 mm [$p = 0.95$] (which was checked several times per day using a vertical graduated scale located near the transducer). Finally, the Lidar was mounted with an incidence angle of $\approx 65°$ and its data were recorded every 10 s.

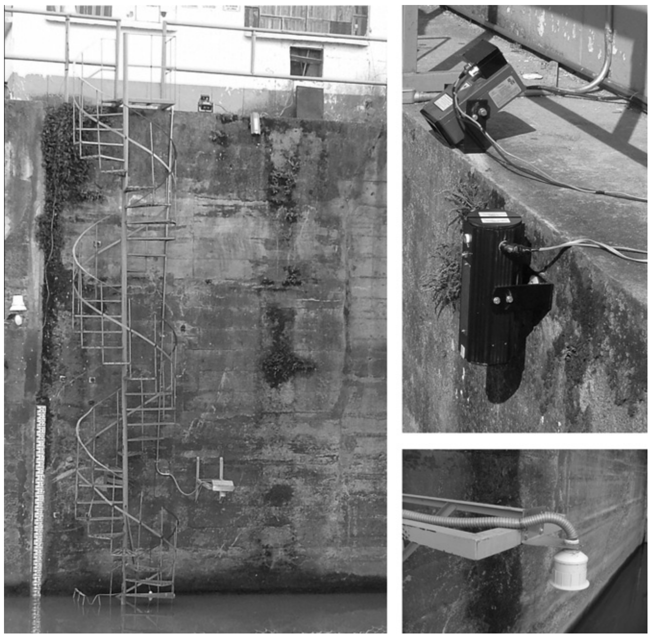
Fig. 2. Experimental set-up in the Tuxpan dam-embankment (Michoacán state, Mexico): (left) view of the dam wall with two Lidar, an ultrasonic water-level transducer, and a vertical graduated scale; (upper right) View of a vertically mounted *Sentry* Lidar (results not shown in this study) and a *Watchman* Lidar with an incidence angle of 65° (results shown in this study); (lower right) View of the ultrasonic water level transducer used as a reference.

3.4. Water-clarity characterization and Lidar data processing

During the tests and the experiment, the water turbidity was assessed using a Secchi disk, which is a black-and-white disk that is lowered into the water by a graduated line. The depth of visual disappearance of the disk – the *Secchi depth* (Z_D, m) – is indeed a simple but useful index of water clarity (Davies-Colley and Smith, 2001).

The raw distance data measured by the Lidar were processed in three steps. First, the data were numerically filtered using two basic criteria: a maximum value for the "dropout number" and a minimum value for the "signal intensity" (the reason why the Lidar data were filtered is explained in Sections 4.1 and 4.2). Then, the distance data (L, m) were converted into stage data (h, m) as: $h = H - L\cos(\theta)$, where θ (°) is the incidence angle of the Lidar, and H (m) is the vertical elevation of the Lidar above the bottom of the considered water body. Finally, the θ and H values used to convert the distance data into stage data were slightly adjusted so that the stage estimated by the Lidar was similar to the stage independently measured at the beginning of a monitoring period. Said adjustment was made within the tolerance of an independent estimation of θ (±3°) and H (±0.02 m).

It is worth noting that the Lidar should theoretically always overestimates the distance to the water surface, because the light emitted penetrates into water (Fig. 1b) and because the light speed in water is smaller. Thus, the adjusted incidence-angle was expected to be always greater than the real Lidar incidence-angle. However, this was not always the case: for instance, the fitted value of θ for the experiment performed in the Tuxpan dam-embankment was 63.85° instead of the 65° originally measured.

4. Results

4.1. Laboratory tests

During the laboratory tests, many raw distance data recorded by the Watchman Lidar were incoherent and erratic when the Lidar was mounted with a large incidence angle over the water tank. Sometimes the measured distance to the water surface was underestimated and, thus, the computed stage was overestimated (e.g., Fig. 3a, at time $\approx$ 50 min). At other times the opposite occurred (e.g., Fig. 3a, at time $\approx$ 60 min), and sometimes the instrument measured nothing, in which case the message "Out of range" was sent by the Lidar.

However, most of the erratic data provided by the Lidar were found to be of poor quality, according to the two available diagnostic parameters of the instrument (see Section 3.1). By trial and error, these "poor quality" data were defined as those for which the "dropout number" was too high (>120) or the "signal intensity" was too low (<10). The remaining Lidar data made it possible to compute stage data that were more consistent with the reference-data (e.g., Fig. 3b).The Lidar was expected to satisfactorily detect the (sub-)surface of the water tank, provided that the water turbidity was high enough (see Section 2.3). By trial and error, a Secchi depth of <0.5 m was found to be the water clarity condition for which the tested Lidar was able to provide continuously reliable stage estimates, within ±0.05 m [$p = 0.95$] (e.g., Fig. 3b, at times <40 min). Considering the stated measurement uncertainty of the tested Lidar (±0.04 m for a diffuse target and a measuring rate of one reading per second, see Section 3.1) and of the instrument used as a reference to monitor the stage (±0.005 m, see Section 3.2), differences of up to $\sqrt{0.04^2 + 0.005^2} \approx 0.04$ m were expected; on this basis, the experimental results obtained in the tank filled with turbid water were therefore quite satisfactory.

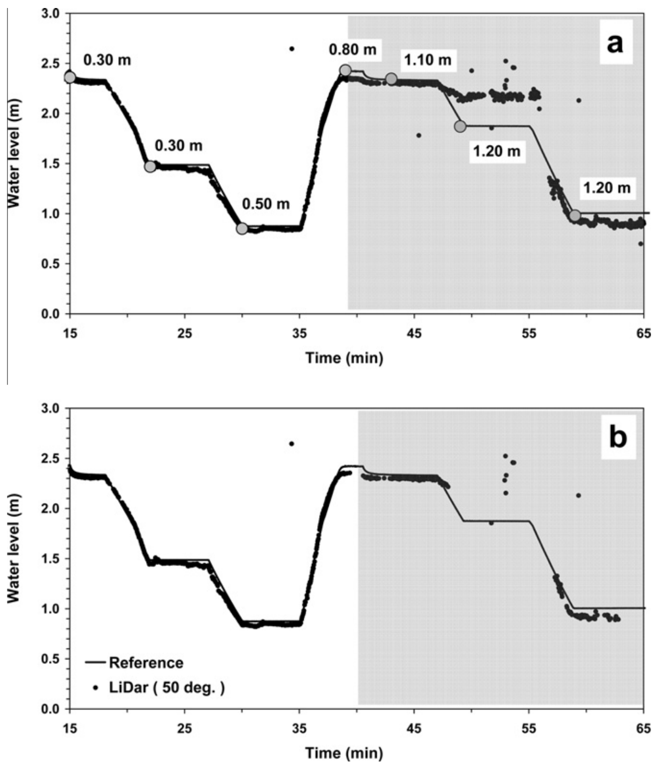

Fig. 3. Results of a laboratory test in which the tested Lidar was mounted with an incidence angle of 50° over a laboratory tank filled with turbid water: (a) non-filtered data and (b) filtered data (see Section 4.1). The measured Secchi depths are shown on the upper plot; the gray area corresponds to less turbid water (Secchi depth greater than 0.5 m).

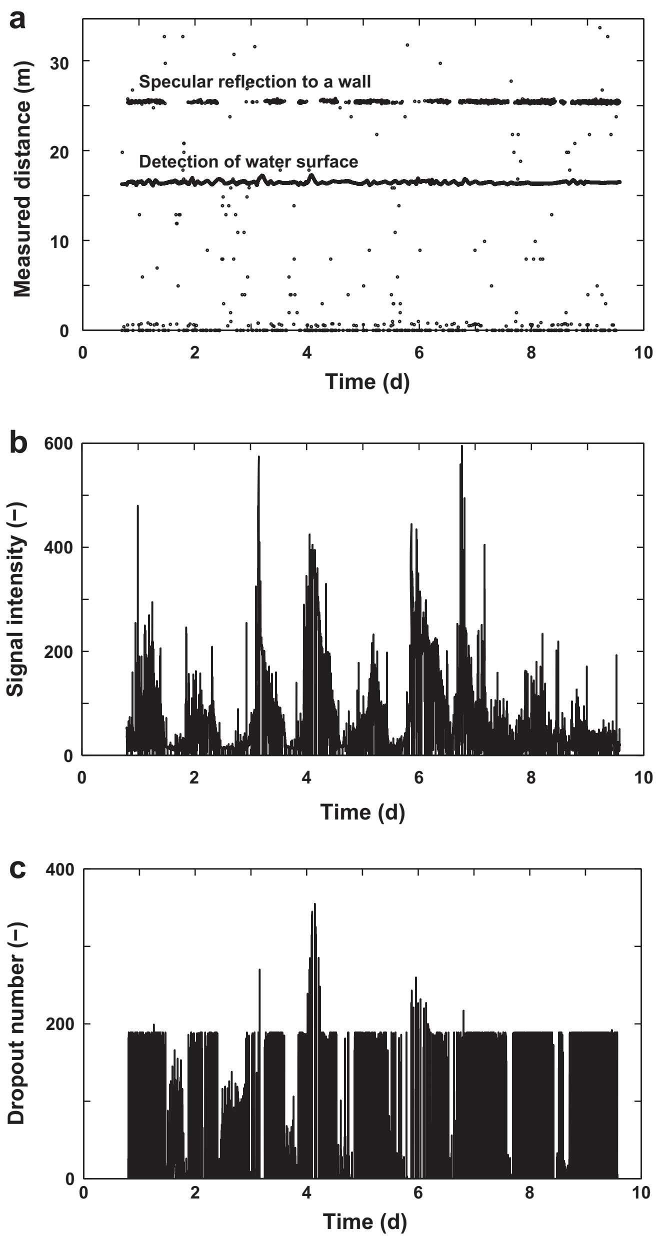

Fig. 4. Raw Lidar data obtained during the field experiment in which the tested instrument was mounted with an incidence angle of 65° over a dam embankment with turbid water (Secchi depth≈0.25 m): (a) measured distances; (b) index of the light intensity received by the Lidar detector; and (c) number of laser shots that do not return to the Lidar during a reading.

When the water was less turbid (Secchi depth > 0.5 m), the tested Lidar was not always able to satisfactorily estimate the stage. Often no "good-quality" data were recorded (e.g., Fig. 3b, at times between 48 and 57 min), or the computed-stage was underestimated by up to ≈0.2 m (e.g., Fig. 3b, at time ≈60 min), or stage data were sometimes overestimated (e.g., Fig. 3b, at time ≈35 and 52 min). The lack of "good-quality" data can be explained by a specular reflection of the light emitted by the Lidar at the water surface and by an absorption of this light by the water without enough backscattering from suspended particles. A stage underestimation of up to ≈0.2 m when the water is less turbid can be explained by an higher penetration of the light emitted by the Lidar into water before it is backscattered by some of the suspended particles; in addition, the distance is also overestimated because the light speed in water is smaller (see Section 2.1). Finally, we do not have an explanation for the overestimated stage data, which correspond to measured distances shorter than the real distance to the water surface. This is surprising because the tested Lidar was configured using the "filtered-last-pulse" mode recommended by its manufacturer to avoid interferences with dust, steam or rain between the instrument and a target (see Section 3.1). Nevertheless; there were only a few overestimated stage data and they were easy to identify when compared to the general trend that was obtained with the Lidar.

4.2. Field experiment

Water turbidity was high in the Tuxpan dam-embankment, with a Secchi depth of ≈0.25 m. As for the laboratory tests (Section 4.1), though raw distance data recorded by the Watchamn Lidar were incoherent and erratic (Fig. 4a), a large amount of data was consistent with the distance to the water surface, which was independently estimated as ≈16.5 m. Another large amount of data gave a distance of ≈26 m, which was explained (e.g., Matthies et al., 2003) by the specular reflection of the light emitted by the Lidar to a vertical wall that was in the other side of the embankment.

During the experiment, the Lidar diagnostic parameters were quite erratic, with a "signal intensity" variation from 0 to ≈600 (Fig. 4b) and a variation in the "dropout number" from 0 to ≈360 (Fig. 4c). Both parameters were more often higher during the night and in the morning. Although we do not know why the "signal intensity" was higher during the night, it suggests that the measuring conditions were improved; in this case, a smaller "dropout number" was expected *a priori*. Surprisingly, the experimental relationship between the "signal intensity" and the "dropout number" was not monotonic (Fig. 5). However, many data corresponding to an high "signal-intensity" (>10) and a large "dropout-number" (>120) were those that were due to an interference with the wall on the other side of the embankment (measured distance ≈ 16.5 m). For the remaining data, a negative correlation between the "signal intensity" and the "dropout number" was obtained (see "good-quality data" in Fig. 5), as expected.

The total precipitation registered during the monitoring period was 28 mm, which is quite small (with six rains between 3 and 7 mm). Therefore, no evidence that the Lidar data were corrupted by rain was found during the experiment; for instance, there was no rain at day 4, when the "dropout number" was suddenly much higher.As expected, most of the erratic distance data provided by the tested Lidar were of poor quality and were filtered in the same way as during the laboratory tests (Section 4.1). That is, the data for which the "dropout number" was >120 or the "signal intensity" was <10 were eliminated. Some unrealistic very large (>24 m) and very small (<2 m) were also eliminated. All the remaining distance data were kept, which represented 84% of all the measured data (≈25,000 in total). After converting the filtered Lidar distance data in stage-data (see Section 3.4), the tested instrument was found to be consistent with the reference stage data (Fig. 6a). On the basis of the reference stage data (842 values), the uncertainty of the Lidar was within ±0.03 m [$p = 0.95$] (Fig. 6b). Considering the stated measurement uncertainty of the tested Lidar (±0.04 m, see Section 3.1) and of the instrument used as a reference to monitor the stage (±0.02 m, see Section 3.3), differences of up to $\sqrt{0.04^2 + 0.02^2} \approx 0.045$ m [$p = 0.95$] were expected; the results obtained in the Tuxpan dam-embankment were therefore satisfactory.

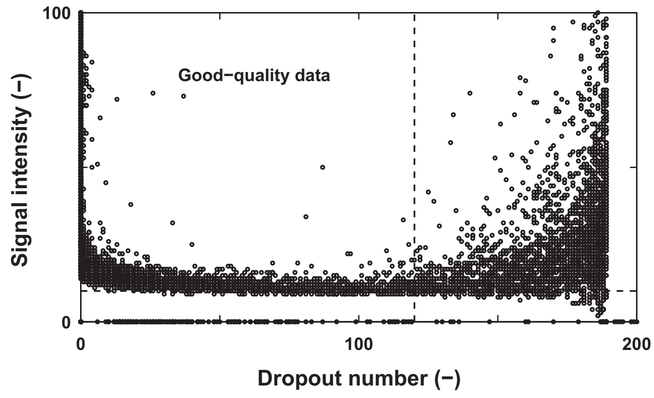

Fig. 5. Part of the relationship obtained between the two diagnostic parameters of the tested Lidar during the field experiment: the "good-quality" data were defined as those for which the "signal intensity" was higher than 10 and the "dropout number" was smaller than 120.

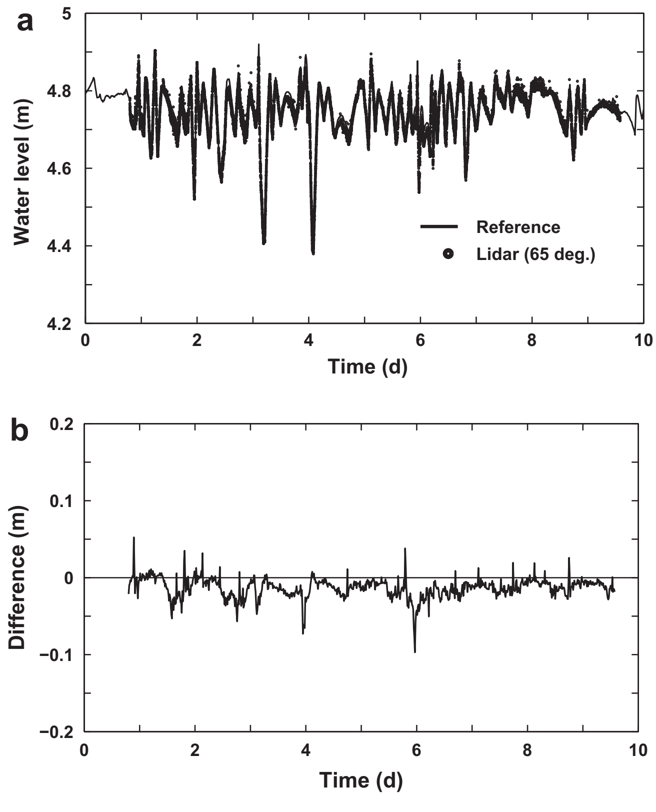

Fig. 6. Processed data obtained from the field experiment in which the tested Lidar was mounted with an incidence angle of 65° over a dam embankment with turbid water (Secchi depth ≈ 0.25 m): (a) water-level data and (b) Lidar level data minus the reference level data.

5. Discussion

5.1. Practical interest of the study

This experimental study shows that a near-infrared Lidar mounted with a large incidence angle (at least between 40° and 70°) can detect the (sub-)surface of turbid water bodies. Therefore, said instrument could be installed at the edge of lakes or earth-dam embankments and be used to monitor the stage. For instance, the instrumentation of an earth-dam embankment would require an instrument mounted with an incidence angle between 63° and 72°, which is in the range of the tested incidence angles. As far as we know, there is no published results about the application described in this study. The cost of the tested Lidar (≈5000 USD) and its ease of installation makes it attractive compared to the cost of installation and maintenance of more traditional techniques, such as a submerged pressure transducer or a bubbler system.

However, it must be recognized that the tested Lidar was only able to satisfactorily detect the surface of very turbid water (Secchi depth < 0.5 m). Water bodies are usually less turbid (e.g., Alsdorf et al., 2007), except for those lakes (e.g., Kent State University, 2010; Bravo-Inclán et al., 2010), reservoirs (e.g., Riera et al., 1992) and rivers (Reeves and Galat, 2010) that present strong problems of eutrophication, land erosion and/or contamination by residual water.

In addition, it must be recognized that the stage-data obtained with the tested Lidar were not very accurate (within ±0.05 m [$p = 0.95$], while the uncertainty for terrestrial traditional techniques to monitor the stage in water bodies is usually within ±0.01 m (IOC, 2006; ISO, 2008) and within ±0.15 m for the new techniques using airborne instruments (e.g., Alsdorf et al., 2007; Höfle et al., 2009; Vuglinskiy, 2009). Therefore, the proposed new-technique should be currently seen as an option for some particular sites where there is no other alternative to monitor the stage.

5.2. Further research

The tested Lidar is a quite simple commercial model that was not built for the kind of application described in this study (see Section 3.1). The versatility (for less-turbid water) and accuracy (detection of a turbid layer closer to the water-surface) of the proposed technique could be improved in the future by using more sophisticated Lidar, i.e. instruments more sensitive to the light backscattered by the particles suspended into water and that can better process such a signal with time. In this context, it is worth noting that the most recent Lidar record the full waveform of the light sent back by a target (e.g., Guenther et al., 2001; Churnside and Donaghay, 2009; Höfle et al., 2009; Allouis et al., 2010).

At least the tested Lidar can be configured to detect the signal sent back by a target with three time-gating options: "direct pulse", "last pulse" and "filtered last pulse" (Optech, 2004). Unfortunately, the instrument firmware did not allowed to use theses options simultaneously, and the last one was chosen because it was recommended by the manufacturer in case of interferences with rain, steam, fog or dust. However, the Lidar firmware could be slightly modified, so that the instrument could take successively readings using each one of the available time-gating option; this would allow a better selection of the "good quality" data, i.e. those for which the successive readings are close.

It has been reported recently that an airborne ultraviolet-Lidar system mounted with an incidence angle up to almost 40° was sensitive to the light backscattered by the particles suspended into the sea (Li et al., 2010). In this case, such an instrument looks attractive for the terrestrial application described in this study, provided that it is configured to detect the first return of the Lidar signal (otherwise, the instrument could detect echoes from particles that are far from the water surface, since the ultraviolet light penetrates deeper into water than the near-infrared light does).

6. Conclusion

A new technique using a near-infrared terrestrial Lidar mounted with a large incidence angle (at least between 40° and 70°) has been proposed to monitor the stage of still but turbid water bodies. This technique could be useful to monitor the stage of some lakes with a sloping bank and some earth-dam embankments. Laboratory tests and a field experiment using a commercial Lidar have shown that the instrument was able to monitor the stage with an uncertainty of about ±0.05 m [$p = 0.95$], provided that the water is very turbid (Secchi depth < 0.5 m). Therefore, the proposed new technique should be currently seen as an option for particular sites where there is no other alternative to monitor the stage. However, the versatility and accuracy of the technique is expected to improve in the future with the use of current Lidar that are more sophisticated than the tested one, i.e. instruments more sensitive to the light backscattered by the particles suspended into water and that can better process such a signal over time. Finally, it is worth noting that a Lidar sensitive to the light backscattered by suspended particles can theoretically determine the velocity of said particles with respect to the instrument (using the Doppler effect); in this case, a set of Doppler Lidar mounted with a large incidence angle could potentially be used to infer (sub-)surface currents in turbid water bodies.

References

Allouis, T., Bailly, J.S., Pastol, Y., Le Roux, C., 2010. Comparison of LiDAR waveform processing methods for very shallow water bathymetry using Raman, near-infrared and green signals. Earth Surface Processes and Landforms 35 (6), 640–650.

Alsdorf, D.E., Rodriguez, E., Lettenmaier, D.P., 2007. Measuring surface water from space. Reviews of Geophysics 45, RG2002, doi:10.1029/2006RG000197 (24pp.).

Bills, B.G., Borsa, A.A., Comstock, R.L., 2007. MISR-based passive optical bathymetry from orbit with few-cm level of accuracy on the Salar de Uyuni, Bolivia. Remote Sensing of Environment 107 (1–2), 240–255.

Bravo-Inclán, L.A., Olvera-Viascán, V., Sánchez-Chávez, J.J., Tomasini-Ortiz, A.C., 2010. Trophic state assessment in warm-water tropical lakes and reservoirs of the central region of Mexico. In: van Bochove, E., Vanrolleghem, P. (Eds.), Proc. 14th International Conference, IWA Diffuse Pollution Specialist Group: Diffuse Pollution and Eutrophication (DIPCON 2010), Agriculture and Agri-Food Canada/IWA, Beaupré, Québec, 12–17 September 2010, pp. 35–39.

Caimi, F.M., Dalgleish, F.R., 2010. Performance considerations for continuous-wave and pulsed laser line scan (LLS) imaging systems. Journal of the European Optical Society – Rapid Publications 5, 10020s, doi:10.2971/jeos.2010.10020s, 10pp.

Carter, W., Shrestha, R., Tuell, G., Bloomquist, D., Sartori, M., 2001. Airborne laser swath mapping shines new light on Earth's topography. Eos Transactions (American Geophysical Union) 82 (46), 549–555.

Churnside, J.H., 2010. Lidar signature from bubbles in the sea. Optics Express 18 (8), 8294–8299.

Churnside, J.H., Donaghay, P.L., 2009. Thin scattering layers observed by airborne lidar. ICES Journal of Marine Science 66 (4), 778–789.

Davies-Colley, R.J., Smith, D.G., 2001. Turbidity, suspended sediment, and water clarity: a review. Journal of the American Water Resources Association 37 (5), 1085–1101.

Doxaran, D., Froidefond, J.M., Lavender, S., Castaing, P., 2002. Spectral signature of highly turbid waters Application with SPOT data to quantify suspended particulate matter concentrations. Remote Sensing of Environment 81 (1), 149–161.

Fulton, J., Ostrowski, J., 2008. Measuring real-time streamflow using emerging technologies: radar, hydroacoustics, and the probability concept. Journal of Hydrology 357 (1–2), 1–10.

Guenther, G.C., Cunningham, A.G., LaRocque, P.E., Reid, D.J., 2001. Meeting the accuracy challenge in airborne Lidar bathymetry. EARSeL eProceedings 1 (1), 1–27.

Heritage, G., Hetherington, D., 2007. Towards a protocol for laser scanning in fluvial geomorphology. Earth Surface Processes and Landforms 32 (1), 66–74.

Höfle, B., Vetter, M., Pfeifer, N., Mandlburger, G., Stötter, J., 2009. Water surface mapping from airborne laser scanning using signal intensity and elevation data. Earth Surface Processes and Landforms 34 (12), 1635–1649.

Huang, N.E., 1981. Survey of remote sensing techniques for wave measurement. In: Flipse, J.E. (Ed.), Measuring Ocean Waves. National Academy Press, Washington, DC, pp. 38–79.

IOC, 2006. Manual on sea-level measurements and interpretation, Volume IV: An update to 2006 (IOC Manuals and Guides No. 14, vol. IV). Intergovernmental Oceanographic Commission/UNESCO, Paris, 78pp.

ISO, 2008. Hydrometry – Water level measuring devices (ISO 4373:2008(E)). International Organization for Standardization, Genève, 26pp.

Kent State University, 2010. The Secchi Dip-In, <http://www.secchidipin.org/> (accessed 8 February, 2011).

Li, Z., Lemmerz, C., Paffrath, U., Reitebuch, O., Witschas, B., 2010. Airborne Doppler Lidar investigation of sea surface reflectance at a 355-nm ultraviolet wavelength. Journal of Atmospheric and Oceanic Technology 27 (4), 693–704.

Matthies, L., Bellutta, P., McHenry, M., 2003. Detecting water hazards for autonomous off-road navigation. In: Gerhart, G.R., Shoemaker, C.M., Gage, D.W. (Eds), Unmanned Ground Vehicle Technology V (Proc. SPIE Conference 5083), SPIE, Bellingham, WA, pp. 231–242, ISBN: 9780819449429.

Menzies, R.T., Tratt, D.M., Hunt, W.H., 1998. Lidar in-space technology experiment measurements of sea surface directional reflectance and the link to surface wind speed. Applied Optics 37 (24), 5550–5559.

Optech, 2004. Watchman 3100 – User Manual (290-000126/Rev B/Aug 04). Optech Inc., Vaughan, Ontario, Canada.

Phillips, D.M., Koerber, B.W., 1984. A theoretical study of an airborne laser technique for determining sea water turbidity. Australian Journal of Physics 37, 75–90.

Reeves, K.S., Galat, D.L., 2010. Do larval fishes exhibit diel drift patterns in a large, turbid river? Journal of Applied Ichthyology 26 (4), 571–577.

Riera, J.L., Jaume, D., de Manuel, J., Morgui, J.A., Armengol, J., 1992. Patterns of variation in the limnology of Spanish reservoirs: a regional study. Limnetica 8, 111–123.

Tamari, S., Laporte-Vergnes, A., Salgado, G., 2010. A simple benchmark to check laser distance-meters. In: Galván-Hernández, C.A. (Ed.), Proc. Simposio de Metrología 2010, CENAM, Querétaro, Mexico, 27–29 October 2010, 7pp. (paper C17 on CD-ROM) [in spanish].

Tamari, S., Aguilar-Chávez, A., 2010. Testing submersible pressure transducers to monitor water level in tanks. Tecnología y Ciencias del Agua 1 (3), 71–88 (in spanish).

Vuglinskiy, V., 2009. Water level – Water level in lakes and reservoirs, water storage (GTOS 59, vers. 8). Global Terrestrial Observing System/FAO, Rome, 18pp.

Zhang, X., Lewis, M., Johnson, B., 1998. Influence of bubbles on scattering of light in the ocean. Applied Optics 37 (27), 6525–6536.

Urban Water Journal
2012, 1–17, i*First* article

Determination of leakage inside buildings with a roof tank

S. Tamari[a]* and J. Ploquet[b]

[a]*Instituto Mexicano de Tecnología del Agua, Jiutepec, México;* [b]*Ecole des Mines de Douai, Douai, France*

(*Received 18 July 2011; final version received 20 January 2012*)

Buildings with a roof tank are common in countries with intermittent water supply. For these buildings, it is difficult to determine the leaks inside with the methods developed for pressurised networks of drinking water. A proposed alternative is *Continuous Automatic Tank Gauging*. Although commonly used to check the tightness of gasoline storage tanks, this method still has not been used to determine water leaks. This study explains how to set up the method and shows its application to six buildings. As expected, the leak magnitude was found to increase with the size and age of buildings. In practice, the method can estimate (within ± 30%) a leak as small as 1 l/h in buildings with a small roof tank and 20 l/h in buildings with a large roof tank.

Keywords: drinking water; intermittent water supply; domestic leakage; Continuous ATG; submersible pressure transducer

1. Introduction

Indoor leaks (i.e., water leaks inside a particular building) are not a priority concern of water suppliers, since they are considered part of domestic consumption (Lee and Schwab 2005, Andey and Kelkar 2009). Nevertheless, they represent higher fees for users. For example, a continuous leak of only 7.5 l/h is equivalent to the average domestic consumption of one person in Mexico, that is, $\approx$ 180 l/d (Grafton *et al*. 2009). Indoor leaks also mean higher electric consumption for those users who need to pump water from a domestic cistern (Trifunovic 2006, Cobacho *et al*. 2008) and they can even cause deterioration of buildings (e.g., when water accumulates in the walls). Finally, indoor leaks can become problematic for the water suppliers themselves; in fact, this loss of water can result in a lack of supply during dry periods (Lambert 2002, Puust *et al*. 2010).

Users traditionally take responsibility for calling a plumber when they detect a leak inside the house. The most common cause of indoor leaks is damaged valves of toilet and tap (Arregui *et al*. 2006). However, when a plumber fixes a leaking device, the leak magnitude is not well known. In addition, there may be hidden leaks (e.g., an embedded pipe that is deteriorated) or leaks so small that they are hardly perceptible. Even though various techniques exist to detect and quantify leaks in drinking water networks, they are not commonly used at the household level. Furthermore, only certain sophisticated techniques—such as Pig-Mounted Acoustic sensing—can detect leaks <10 l/h (Puust *et al*. 2010).

In industrialised countries, water supply is generally *continuous*, that is, the drinking water network is always pressurised, and thus users usually do not need to store water in domestic tanks (Figure 1a). In this case, the magnitude of indoor leaks can be estimated by analysing data provided by the domestic water meter (Cobacho *et al*. 2008, Willis *et al*. 2009). Simulation models that analyse water pressure or the concentration of a tracer measured at different nodes of the water network can also be used (Puust *et al*. 2010).

In countries with few water and/or economic resources, water supply is generally *intermittent*, that is, the drinking water arrives at the domestic intakes only during certain hours of the week (Lee and Schwab 2005, Trifunovic 2006). This situation results in many users storing water in domestic tanks (Figure 1b). In this case, it is very complicated to estimate indoor leaks based on the domestic water meter data (Arregui *et al*. 2006, Cobacho *et al*. 2008, Criminisi *et al*. 2009) or with simulation models (in fact, this has not yet been attempted). Therefore, little is known about the magnitude of indoor leaks in buildings with intermittent water supply and storage tanks. One of the most commonly used tanks in Latin America,

*Corresponding author. Email: tamari@tlaloc.imta.mx

ISSN 1573-062X print/ISSN 1744-9006 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/1573062X.2012.660959
http://www.tandfonline.com

South Asia, Middle East, Mediterranean countries and Africa is a *roof tank*, because it maintains the plumbing within a building pressurised (Trifunovic 2006, Andey and Kelkar 2009, Criminisi *et al*. 2009). Therefore, the objective of this study has been to identify and test a method to determine leaks inside buildings with a roof tank.

2. Fundamentals

2.1. Options for determining leaks in buildings with a roof tank

2.1.1. Install a water meter at the tank outlet

To estimate leaks inside a building with a roof tank, a water meter with a data logger can be installed at the tank outlet and its data analysed (Arregui *et al*. 2006, Cobacho *et al*. 2008, Criminisi *et al*. 2009). Nevertheless, this is not always practical. Installing a conventional water meter requires emptying the tank and cutting the pipe at the outlet, while installing a clamp-on ultrasonic flow meter requires a sufficiently long portion of exposed pipe. Though an alternative would be to install several water meters in different strategic places inside the building, such as toilets, washing machines and showers (Arregui *et al*. 2006, Otaki *et al*. 2008), this solution is expensive and not discreet.

In addition, common water meters are often inappropriate to estimate leaks inside buildings with a roof tank, because they cannot measure accurately small flows in pipes with little pressure (Arregui *et al*. 2006, Criminisi *et al*. 2009). For instance, Figure 2 shows an estimation of the indoor leaks obtained using four water meters (model "FAM-20", Arad™, Dalia, Israel) installed at the outlet of a large roof tank ("Building II" case, described in Section 3.1). As can be seen, on several occasion estimations less than 5 l/h were obtained, which is low for a 15 year-old building with 16 apartments. Such a low value is probably due to the limited sensitivity of the meters used at low flow ("multijet" type, "permanent flow rate (Q_3)" = 2.5 m^3/h and "Q_3/Q_1 ratio" = 50); in fact, they systematically underestimate flows smaller than their "minimum flow rate (Q_1)" = 50 l/h (Lambert 2002, Arregui *et al*. 2006). It must be recognised that meters of the same accuracy but with a smaller "permanent flow rate (Q_3)", for instance 1.5 m^3/h, would have been more suitable in this case. Nevertheless, using water meters to determine indoor leaks can be challenging: it is difficult indeed to find a meter that can accurately measure flows <10 l/h (Lambert 2002, Arregui *et al*. 2006, Trifunovic 2006, Criminisi *et al*. 2009), whereas leaks

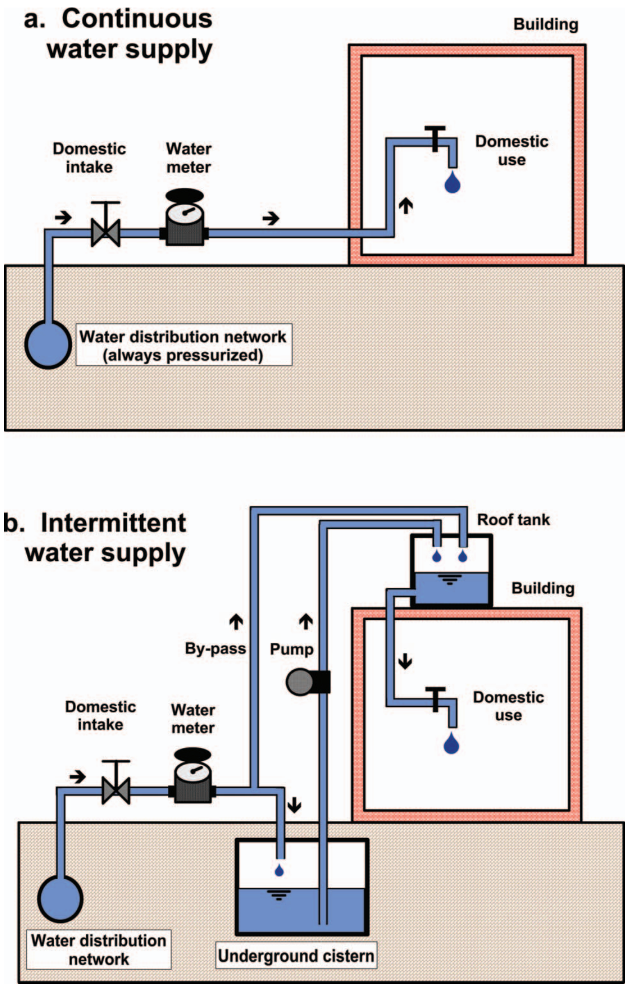

Figure 1. Sketch of buildings with (a) continuous and (b) intermittent drinking water supply.

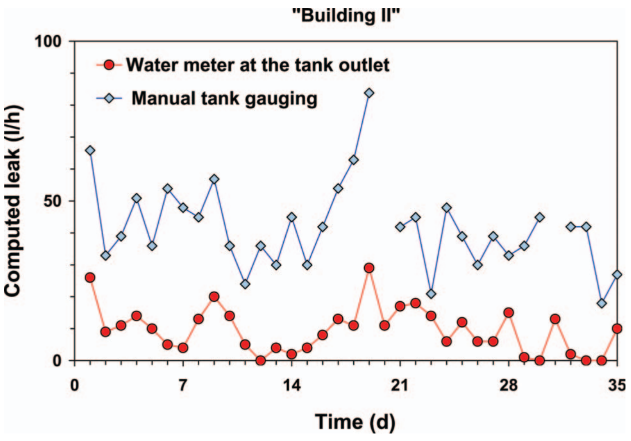

Figure 2. Two ways to estimate water leaks inside a building with a roof tank (example): minimum hourly consumption detected each day by water meters installed at the outlet of the tank (circle) and decrease in volume in the tank observed each day between 1:00 and 5:00 (diamond).

in private houses are often < 6 l/h (Arregui *et al.* 2006, Willis *et al.* 2009).

2.1.2. *Manual measurement of water level in the roof tank*

Another option to estimate leaks inside a building with a roof tank is to measure the decrease in the water level inside the roof tank when there is no intentional consumption or supply. By knowing the internal geometry of the tank, it is easy to convert the change in level to a change in volume, and thus infer the indoor leak. This method—known as *Manual Tank Gauging*—is the most traditional method to detect leaks in gasoline storage tanks (EPA 2005). Nevertheless, its strict application to a building with a roof tank is not practical because the method requires cutting off the supply and requesting users not to consume water during various hours.

As an alternative, it is possible to consider that the users *generally* do not intentionally consume water at night, that is, between ≈1:00 and 5:00 (Andey and Kelkar 2009, Puust *et al.* 2010). In this case, leaks in a building with a roof tank can be estimated by measuring the decrease in the water level in the tank between two different hours of the night. Nonetheless, this simple method has two obvious disadvantages (in addition to being laborious): it will underestimate the leak if the tank suddenly begins to fill, and overestimate the leak if someone consumes water during the night. For instance, Figure 2 shows an estimate of indoor leaks based on the decrease in water from the roof tank between 1:00 and 5:00; no elements exist to know whether or not the leak estimates (from 20 to 80 l/h) are reliable (in addition to not being able to estimate a realistic value for leaks on two days, because the roof tank was filling).

2.1.3. *Monitoring the water level in the roof tank*

A refinement of the above method consists of continually monitoring the water level in a roof tank, converting the changes in the level to changes in stored volume, and automatically processing the data to identify the slowest decreases in volume, which will be interpreted as leaks. This method—known as *Continuous Automatic Tank Gauging*—has been used for 20 years to check the tightness of gasoline storage tanks (Flora 2000, EPA 2005, NWGLDE 2007). As far as we know, it has never been used to estimate water leaks. This is probably due to economic issues: on the one hand, there is greater concern about gasoline leaks than domestic water leaks and, on the other hand, legislation to detect gasoline leaks (as little as 0.8 l/h in tanks of up to 75 m^3) is strict, and therefore requires sophisticated and expensive detection systems. Thus, this study has focused on identifying a simple and economic manner to implement the "Continuous Automatic Tank Gauging" method in buildings with a roof tank and intermittent water supply. An algorithm is proposed to process the data, because none has been found in the scientific literature.

2.2. *Scope of the study*

Considering that a leak is any unwanted water consumption, it would be complicated to detect all types of leaks. For instance, when a user leaves a tap open in a building, it is difficult to know whether or not it is intentional. Therefore, the scope of this study is to quantify only those leaks that represent the minimal water consumption in a building. For a building with a roof tank, these leaks should be apparent at certain hours of the day—when people are asleep or out—by a slow and regular decrease in the volume of water stored in the roof tank.

Roof tanks are not exclusively used in areas of intermittent water supply: they can be also used in areas of continuous supply, when the drinking water network is not pressurised enough or when its transportation capacity is too low; however, this situation is out of the scope of the study. Next, buildings with a roof tank are considered that have the following characteristics: (1) water supply is intermittent (something that users would undoubtedly know), (2) the roof tank works satisfactorily (that is, it does not normally empty completely: something that users would also undoubtedly know), (3) the roof tank does not fill too slowly (something that can be checked *a posteriori*, as explained in Section 3.2.5) and (4) there are periods of the day when water is normally not consumed (in this case, buildings such as hospitals and large factories that operate all the time are discarded).

This study considers roof tanks with a storage capacity between ≈ 0.5 and 15 m^3, typical for buildings with 1 to ≈100 inhabitants. In the following, the size of roof tanks is classified according to their cross-sectional area (A): "small" tanks are those for which $A \approx 0.5$ m^2, "medium" tanks are those for which $A \approx 2$ m^2 and "large" tanks are those for which $A \approx 10$ m^2.

3. Materials and methods

3.1. *Buildings where the proposed method was tested*

This study was conducted in Mexico. Six cases were considered to test the proposed method (Table 1). The first ("Almost empty house") is a residential house with a small roof tank (Figure 3a) and that

Table 1. Description of the monitoring tests in buildings with a roof tank.

Test	Case	a	b	c	d	e	f
	Name	"Almost empty house"	"Occupied house"	"Office"	"Building 0"	"Building I"	"Building II"
	Period	28 d (22/04 - 21/05, 2010)	23 d (09/09 - 03/10, 2007)	24 d (17/03 - 11/04, 2008)	34 d (24/08 - 28/09, 2008)	36 d (06/07 - 12/08, 2010)	
Roof tank	Origin	Plastic ("Rotoplas", Mexico City)			Brick and cement		
	Shape	Cylinder			Parallelepiped	Prismatic (the cross-section is an irregular hexagon)	
	Nominal capacity (m^3)	0.5		2.5	16	12	
	Cross-section area (m^2)	0.567		1.95	10.8	11.6	11.7
	Filling method	Automatically pumped from a cistern and *by-pass* from the domestic intake	Same as case (a), but with less difference between operating levels (for test purposes)	Automatically pumped from a cistern	Semi-automatically pumped from a cistern and *by-pass* from the domestic intake	Pumped during certain hours of the day from a cistern	Same as case (e), but with more flow during tank-filling (because of the site topography)
Building	Location	Cuernavaca (Morelos)		Jiutepec (Morelos)	Mexico City (DF)	Jiutepec (Morelos)	
	Age (years)	5	2	10	30	15	
	Users	1 (1 adult)	3 (2 adults +1 child)	64 (60 civil servants + 4 cleaning staff)	48 (44 adults + 4 children)	42 (28 adults + 4 children)	51
	Social category	Upper middle class (income ≈ 2500 USD per month, per family)		Administrative building	Upper middle class (income ≈ 2900 USD per month, per family)	Lower middle class (income ≈ 1200 USD per month, per family)	

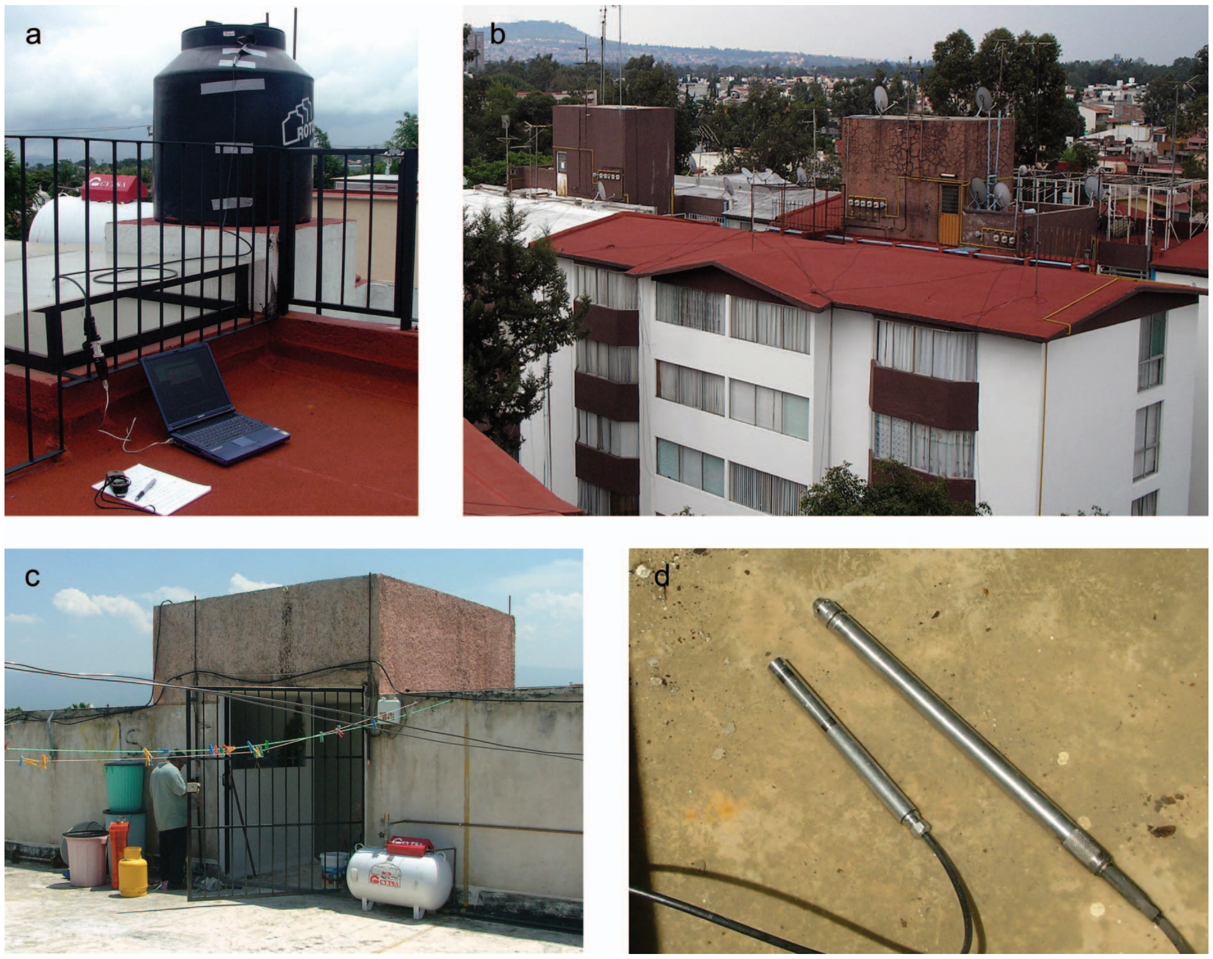
Figure 3. Leak determination in buildings with a roof tank: (a) roof tank on the ''Almost empty house'' and the ''Occupied house,'' (b) roof tank on ''Building 0'', (c) roof tank on ''Building I'' and ''Building II'' and (d) instruments to monitor the water level in the roof tanks.

was not occupied for most of the day during the week. The second (''Occupied house'') is the same case, but when occupied by three people. The third case (''Office'') is an administrative building with a medium-sized roof tank. The other cases (''Building 0'', ''Building I'' and ''Building II'') are private buildings with a large roof tank and ≈16 apartments (Figures 3b,c).

As explained in the following, the water level in the roof tank of each building was monitored (Figure 3d) for ≈4 weeks and with a time step of 1 min. At the end of each monitoring test, the raw data were processed with a code based on the algorithm described below (Section 3.2). This code (available upon request from the authors) was written in ''Matlab'' (The Math Works Inc. ™, Natick, MA). The calculations with a laptop took <2 min. per case for a preliminary (non iterative) leak estimation, and <2 h for an optimised (iterative) estimation.

3.2. *Implementation of the proposed method*

In theory, the magnitude of a leak depends on water pressure (Arregui *et al.* 2006, Puust *et al.* 2010). Nevertheless, this effect was thought to be small, because the height of a building is generally large relative to that of a roof tank. Therefore, the basic idea of the proposed method is that a leak from a roof tank translates as a slow decrease in stored volume, which varies almost linearly with time, and can be observed over sufficient time during certain hours of the day. Eight steps are proposed for the implementation of the method (Figure 4).

3.2.1. *Monitoring the water level in a roof tank*

To monitor the water level (h) in roof tanks, the use of recently available instruments (on the market for ≈15 years) is proposed, that are composed of a submersible pressure transducer (gauge type), a

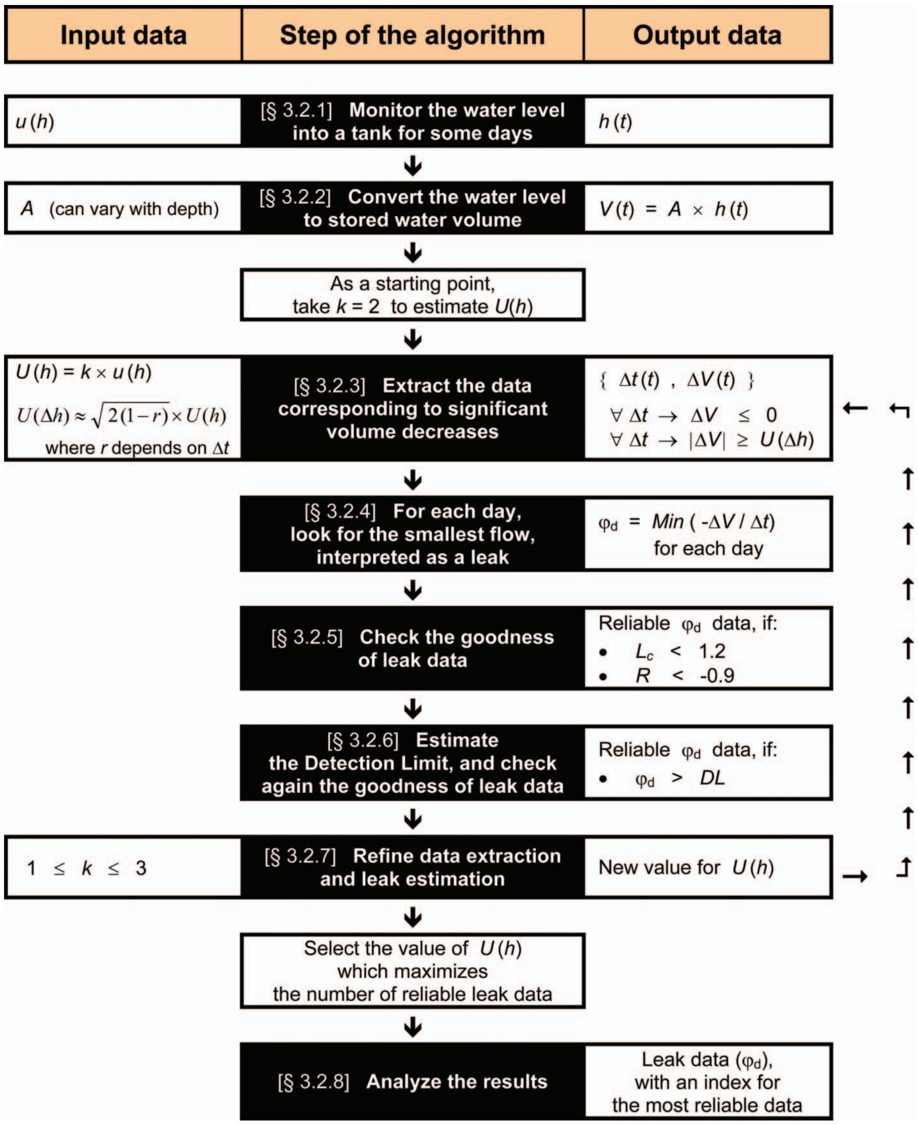
Input data
Step of the algorithm
Output data
u (h)
[§ 3.2.1] Monitor the water level into a tank for some days
h (t)
A (can vary with depth)
[§ 3.2.2] Convert the water level to stored water volume
V (t) = A × h (t)
As a starting point, take k = 2 to estimate U(h)
U (h) = k × u (h)
U(Δh) ≈ √(2(1−r)) × U(h)
where r depends on Δt
[§ 3.2.3] Extract the data corresponding to significant volume decreases
{ Δt (t) , ΔV (t) }
∀ Δt → ΔV ≤ 0
∀ Δt → |ΔV| ≥ U (Δh)
[§ 3.2.4] For each day, look for the smallest flow, interpreted as a leak
φd = Min (-ΔV / Δt) for each day
[§ 3.2.5] Check the goodness of leak data
Reliable φd data, if:
• Lc < 1.2
• R < -0.9
[§ 3.2.6] Estimate the Detection Limit, and check again the goodness of leak data
Reliable φd data, if:
• φd > DL
1 ≤ k ≤ 3
[§ 3.2.7] Refine data extraction and leak estimation
New value for U (h)
Select the value of U (h) which maximizes the number of reliable leak data
[§ 3.2.8] Analyze the results
Leak data (φd), with an index for the most reliable data

Figure 4. Algorithm of the proposed method. Each step is referred to a given section, and the symbols are defined into the text.

temperature sensor (to compensate for the thermal sensitivity of the gauge) and an integrated data logger (with batteries). These can monitor the level in a tank of drinking water for weeks, with a standard uncertainty $u(h)$ better than ± 2.5 mm $[p = 0.68]$ (Tamari and Aguilar-Chavez 2010). This is quite satisfactory as compared to the legal requirements for monitoring the level of liquid in storage tanks. In fact, the maximum permissible error is currently required to be ± 4 mm (OIML 2008). In addition, the proposed instruments are compact (diameter < 20 mm, length < 250 mm), can store thousands of data ($> 28{,}000$ records) and are quite inexpensive ($<$ \$ 1500 USD). Therefore, this is currently a good balance between accuracy, ease of use and cost.

The instruments used were model "Kpsi-551" (Pressure Systems, Hampton, USA) or model "DCX-16VG" (Keller, Newport News, USA), with a full scale between 30 kPa and 50 kPa (Figure 3d). At the beginning of each monitoring test, a pressure transducer was introduced into the roof tank (gently shaken in the water to purge it) and laid on the bottom of the tank (in practice, it is not important that the instrument measures exactly the water level, what matters is that it estimates the *changes* in level with good accuracy). A plastic cap was placed on the end of the transducer cable (to protect it from bad weather).

3.2.2. *Converting level changes into changes in volume*

Knowing the water level in a tank (h, m), the stored water volume (V, m^3) can be computed provided that the internal tank geometry is known. Because many roof tanks have vertical walls (at least within the range of their operating levels), their cross-sectional area (A) is constant. In this case, it simply follows that $V = A \times h$. In addition, many small or medium roof tanks have a circular cross-section (Figure 3a). In this case, it is easy to determine the value of A by measuring the external perimeter and the wall thickness of the tank with a flexible tape. It is also easy to determine the value of A for larger roof tanks with a rectangular cross-section (Figure 3b). Finally, it must be recognised that determining the internal geometry of roof tanks with a more complicated shape can be laborious. Nonetheless, when a tank is large enough for someone to go inside (Figure 3c), the determination of A can still be achieved by triangulation (using Heron's formula) in less than one hour. This requires measuring the length of all the diagonals inside the tank, which can be done with an uncertainty of ± 2 mm [$p = 0.95$] using a handheld laser distance meter (Tamari *et al.* 2010).

When possible, it is desirable to check the stored volume estimations. For small and medium roof tanks, this can easily be done in less than 2 hours. Check consists of producing abrupt changes in the water level in the tank by opening a tap (when nobody is consuming water) and measuring the volume of water coming out from the tap using a calibrated flask. Using this method, differences $<2\%$ were obtained between estimated and measured volume for roof tanks with a capacity of up to 2.5 m^3 (Figure 5). Nevertheless, it must be recognised that it is not easy to experimentally check the volume estimations for larger roof tanks (in particular, because it is difficult to reach agreement with the users to not consume water for $\approx$2 hours).

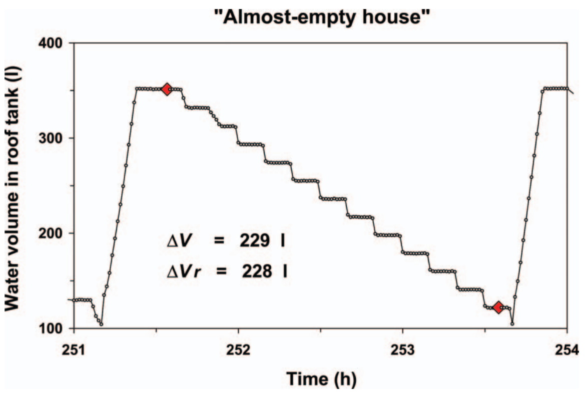

Figure 5. Check of the stored volume estimations for a roof tank (see Section 3.2.2): ΔV_r is a change in volume measured with a volumetric flask (19.0 l), while ΔV is the same change estimated by the proposed method. The diamonds represent the beginning and end of check.

3.2.3. *Extracting data*

Generally, tank-filling (from an underground cistern and/or domestic intake) has the following characteristics: at a scale of a few minutes, water flow from tank-filling is higher than the *average* of water flows from consumption (otherwise, there would be obvious problems with satisfying water demand, i.e., the tank could sometimes become empty, and the users would undoubtedly know). In this case, temporal changes in water level inside a roof tank—that is, the $h(t)$ relationship—can be divided into two categories: periods during which the level increases obviously corresponds to tank-filling periods (and eventual consumption) and other periods during which level decreases *probably* correspond to periods of consumption only.

Our experience is that the periods of decreasing water volume in a roof tank can be automatically identified using some empirical but simple rules: (1) define the local minima of the $h(t)$ relationship as any abrupt increase in water level (e.g., when the water level decreases at least once during three previous time-steps and then increases during at least the following two time-steps), (2) define the local maxima as the maximum water level located between two consecutive local minima, (3) redefine all the local minima as the lowest minimum water level registered between two consecutive local maxima. Finally, it is useful to plot the $h(t)$ relationship and observe the location of the local minima and maxima to check the goodness of the procedure. When water level data are noisy, it can be helpful to smooth them before trying to automatically detect the periods of decreasing water volume; otherwise, these periods can still be manually identified by looking at the plot of the $h(t)$ relationship.

After identifying the periods during which the water level in a roof tank decreases, it is proposed to extract the data that correspond to successive changes in water level that are statistically significant; in practice, these changes will correspond to time intervals (Δt) much larger than the time step used to acquire the raw $h(t)$ data (1 min.). Considering a level reading (h_1) obtained for a certain time (t_1) and another reading (h_2) obtained later ($t_2 = t_1 + \Delta t$) with the *same* instrument, the difference between readings, $\Delta h = |h_1 - h_2|$, will be significant if greater than $U(\Delta h) \approx \sqrt{2(1-r)} \times U(h)$, where $U(h)$ is the expanded uncertainty of the water level data and r ($-1 \leq r \leq 1$) is the correlation coefficient between the readings (JCGM 2008). The expanded uncertainty is: $U(h) = k \times u(h)$, where $u(h)$ is the standard uncertainty of the water level data (Section 3.2.1) and k is the coverage factor (JCGM 2008).

In practice, the response of instruments to monitor water level always depends somewhat on ambient temperature, and at the scale of weeks, the temperature typically varies with a periodicity of 24 hours. In this case, the correlation coefficient (r) tends to be a function of the interval between the two readings (Δt), as shown in Figure 6. For the instruments used in this study (Section 3.2.1), the relation between r and Δt was roughly modelled as: $r = 0.5$ when $\Delta t \leq 2$ h, $r = 0$ when $2 < \Delta t \leq 6$ h, and $r = -0.8$ when $\Delta t > 6$ h (see the bold line in Figure 6); the model intentionally underestimates the experimental values of r, to ensure that the $U(\Delta h)$ criterion will be always maximised. Taking a coverage factor $k = 2$ as a starting point (which would correspond to a confidence interval of $p = 0.95$), the following rule was therefore used to extract the data corresponding to significant water level changes: $U(\Delta h) = 5$ mm when $\Delta t \leq 2$ h, $U(\Delta h) = 7.5$ mm when $2\text{ h} < \Delta t \leq 6$ h and $U(\Delta h) = 10$ mm when $\Delta t > 6$ h. Nevertheless, this rule can be refined for each case under consideration, with an optimisation of k (and thus of $U(h)$); this will be seen below (Section 3.2.7).

3.2.4. *Estimating leaks on a daily basis*

Once the data corresponding to significant changes in water level –and thus in stored water volume– are extracted, water flows in a roof tank can be estimated as: $\varphi = -\mathrm{d}V/\mathrm{d}t \approx -\Delta V/\Delta t$, where ΔV is a significant change in water volume in the tank and Δt is the associated time interval. The sign convention used is a positive φ-value to correspond to a decrease in stored water volume in the tank.

Finally, an indoor leak (φ_d) can be estimated as the minimum value of positive water flows calculated during a day (arbitrarily taken between 0:00 and 24:00 hours). Although the proposed flow calculation is a coarse numerical approximation to estimate most of the domestic water consumption pulses (because their duration is usually less than a minute; Trifunovic 2006), it is theoretically adequate to estimate leaks (because they are almost constant flows at the scale of hours).

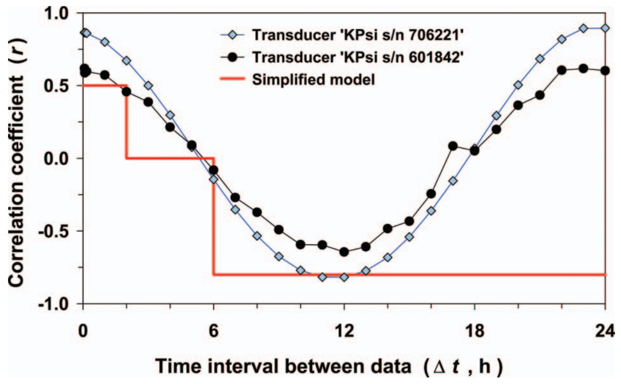

Figure 6. Correlation coefficient (r) between two readings taken with the same pressure transducer, as a function of the time interval between the readings (Δt). Two experimental correlograms are shown (computed using data shown in Figure 4 of Tamari and Aguilar-Chávez 2010). The bold line shows the simplified model used to compute leaks (see Section 3.2.3).

3.2.5. *Checking the goodness of leak estimations*

After estimating indoor leaks on a daily basis (φ_d), it is proposed to check the goodness of these estimates as follows: for each period during which a leak has been estimated (Δt, Sections 3.2.3–3.2.4), review *all* the stored volume data (V, Section 3.2.2) and check that they decrease linearly as a function of time. It is not easy to show that a trend is linear. Two statistical tests are proposed for this purpose:

- *Hansen's linearity test (1992)*. This test is to check that the $V(t)$ relationship does not present irregularities. To accomplish this, the *joint stability test statistic* (L_c) should be calculated, as explained by Hansen (1992). If L_c is greater than a certain value, the hypothesis of linearity is probably not true. According to Hansen (1992), the hypothesis of linearity is probably not true when $L_c > 1.01$ [$p = 0.95$] or when $L_c > 1.35$ [$p = 0.99$] (regardless of the number of data for the experimental relationship under consideration). In this study, an experimental $V(t)$ relationship was considered as linear when $L_c < 1.2$. The linearity test should be especially useful when the time interval (Δt) for which a leak estimation is obtained is large. In this case, the $V(t)$ relationship could present non-linearity due to domestic consumption pulses or the occurrence of a slow tank-filling.
- *Test of the linear correlation coefficient*. This test is to check that the data for the $V(t)$ relationship decrease significantly. Assuming that the decrease is linear (see previous test), this will be true if the linear correlation coefficient (R) of the relationship is close to -1. In this study, an experimental relationship $V(t)$ was considered to be significantly decreasing when $R < -0.90$ (which corresponds to $R^2 \approx 0.80$, that is, more than 80% of the variance in the $V(t)$ relationship is explained by the linear regression model). The test of the correlation coefficient should be especially useful when the time interval (Δt) for which the leak is estimated is small; in this case, the $V(t)$ relationship could end up being too "noisy".

According to a visual analysis of the data, leak estimations are reliable when the two previously mentioned criteria are met (that is, $L_c < 1.2$ and $R < -0.90$); otherwise, they are questionable. One common cause is noisy data, and another is over-estimation due to domestic consumption pulses (Figure 7a). Another less common reason is under-estimation due to the occurrence of a slow tank-filling (Figure 7b): on the one hand, a roof tank connected to a bypass may start to fill slowly, when pressurised water arrives at the domestic intake; on the other hand, a tank connected to an underground cistern may end to fill slowly, when its inlet valve to prevent overflow does not close immediately.

3.2.6. *Estimating the method's Detection Limit*

To determine whether the leak estimations are realistic, it is also useful to estimate the method's Detection Limit, that is, the smallest flow that can be identified. In fact, if a leak is less than this limit, it will tend to be overestimated. The Detection Limit is: $DL = A \times U(\Delta h)/\Delta t_{\max}$, where A is the area of the roof tank, $U(\Delta h)$ is the significant difference between two level readings and $\Delta t_{\max}$ is the maximum duration for which there is generally no intentional water consumption in a building.

As a starting point, if significant changes equal to $U(\Delta h) \approx 10$ mm are detected (Section 3.2.3), the minimum change in stored volume that could be detected is 5 l for a small roof tank ($A = 0.5$ m^2), 20 l for a medium tank ($A = 2$ m^2) and 100 l for a large roof tank ($A = 10$ m^2). Thus, assuming that a period of up to $\Delta t_{\max} \approx 5$ h with no intentional water consumption occurs in a building each day (Section 2.1.2), it is deduced that $DL \approx 1$ l/h for a small roof tank, ≈ 4 l/h for a medium tank and ≈ 20 l/h for a large roof tank (for these Detection Limits, and considering that evaporation from a nearly closed tank is < 3 mm/d, it can be demonstrated that the effect of evaporation is negligible).

As will be seen below, the calculation of DL can be refined if more realistic values of A, $U(\Delta h)$ and $\Delta t_{\max}$ are determined for each case under consideration. Nevertheless, DL is not intended to be estimated with great accuracy, but rather to obtain an order of magnitude so as to observe the goodness of the leak estimations.

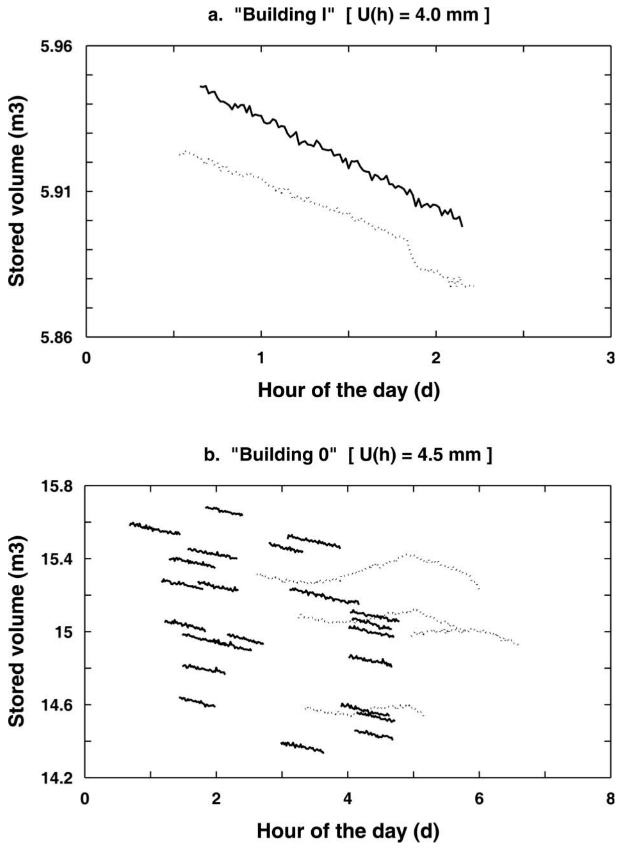

Figure 7. Check of the leak estimations (see Section 3.2.5): automatic identification of reliable data (continuous lines) and unreliable data (dotted lines). Unreliable data shown on the upper graph are due to a domestic consumption pulse, whereas unreliable data shown on the lower graph are due to a roof tank that fills slowly at the beginning.

3.2.7. *Refining leak estimations*

After having checked the goodness of the leak estimations on a daily basis (Section 3.2.5), the number of reliable estimations obtained (N) is known and the average value of these estimations ($\varphi^{\bullet}$) can be calculated. At this time, it is useful to refine the leak estimation with an optimisation of the parameter $U(h) = k \times u(h)$. In fact, the determination of leaks with the proposed method not only depends on $u(h)$, that is, the standard uncertainty of the instrument used to monitor the water level (Section 3.2.1) but also, to a certain extent, on the magnitude of the leaks and the water consumption pattern of the building. For a large roof tank, it is more difficult to distinguish between a leak (signified by a slow and regular decrease in the water depth) and intentional water consumption (signified by a more abrupt decrease in the water depth); therefore, it will be easier to identify a leak by processing data sets that correspond to greater reductions in water level (up to a certain limit).

It is proposed to optimise the value of $U(h) = k \times u(h)$ as follows: (1) consider a range of possible values for the parameter, (2) determine the leaks with each one of these values (that is, follow the steps described in Sections 3.2.3–3.2.6) and (3) choose the optimum value of $U(h)$ as that for which the maximum number of reliable leak estimations (N) is obtained. For a standard uncertainty $u(h) = 2.5$ mm (Section 3.2.1), values of the expanded uncertainty $U(h)$ between 2.5 mm [$k = 1$, $p = 0.68$] and 7.5 mm [$k = 3$, $p = 0.99$] can be considered.

Obviously, the average value of the leak estimations ($\varphi^{\bullet}$) should tend to increase with the value of the $U(h)$ parameter. In fact, with a small $U(h)$ value, there is more likelihood of finding slow decreases in the $V(t)$ relationship, which can be interpreted as "leaks" (Section 3.2.4). On the other hand, with a large $U(h)$ value it will be more difficult to detect small leaks (Section 3.2.6). The statistical tests to assess the goodness of each leak estimation (Section 3.2.5) are intended to stop the trend of the $\varphi^{\bullet}$ value to increase with the $U(h)$ value; nevertheless, these tests *cannot* totally prevent this increase. Therefore, it is important to choose a value of $U(h)$ that is realistic—not too small or too large—in order to adequately estimate indoor leaks.

3.2.8. *Analyse the results*

At the end of data processing, it is first useful to visually review the data using two graphs. One corresponds to the variation in the water level (h) in a roof tank as a function of time (t) in order to know whether the roof tank is operating adequately, that is, it does not empty or overflow. The other graph is the variation in stored volume (V) as a function of the hour of the day so as to know whether the hours of the day for which leaks are estimated are consistent with what is expected, that is, a slow and regular decrease in the level in the roof tank during hours of the day for which water is not intentionally consumed.

It is then useful to analyse the results of the optimisation of $U(h)$ to see whether a realistic value exists for this parameter that allows for estimating leaks inside a given building. Finally, it is interesting to analyse the variability in the leaks estimated on a daily basis during the monitoring campaign; such a variability could reveal indeed some trends, such as differences between the week days and the weekends, or a change in the plumbing of the studied building.

3.2.9. *Comment about data processing*

The data processing algorithm attempts to estimate a leak *only once* for each day of a monitoring test. It could be argued that when this estimation turns out not to be reliable (Section 3.2.5), it would be useful to refine data analysis at the scale of a day, to see if a reliable leak value can be obtained. Nevertheless, this would involve longer calculations (e.g., using a Genetic Algorithm). In addition, our experience is that the leak estimations obtained with the proposed algorithm are commonly rejected because they are noisy data or have had domestic consumption pulses during the period of time required to identify a leak (Figure 7a). Therefore, if an estimation of a reliable leak for a certain day is not achieved the first time, it seems unlikely that a reliable leak will be able to be detected for this same day with a more sophisticated algorithm.

4. Results and discussion

4.1. *Analysis of raw data*

4.1.1. *Water level as a function of time*

The changes of water level observed during the monitoring campaigns show that the roof tanks of the studied buildings work satisfactorily, i.e. they do not completely empty and never overflow (Figure 8). In addition, the data were consistent with what was known about the operation of the roof tanks (Table 1): on the one hand, the water level in the tanks were generally between two extreme values for the first three buildings (Figure 8a–c) due to regulation by an electric level switch (the main exceptions were due to intermittent supply from the domestic intake through a bypass in the case of the "Occupied house", water pumps that started systematically after brief power outages in the case of the "Almost empty house", and a long power outage in the "Office"); on the other hand, the water level was generally less than a certain value for the roof tanks in the other three studied buildings (Figure 8d–f) due to a float-valve control (more or less adequately adjusted), and the minimum levels reached on a daily basis were highly variable due to a lack of regulation (in fact, the roof tanks of the buildings were almost always being filled during the same hours of the day, regardless of the stored volume).

4.1.2. *Stored volume as a function of hour of the day*

The relationship between the stored volume in roof tanks and the hour of the day reveals various trends (Figure 9), one of which is increases in volume that are obviously due to water supply events. As expected (Table 1), these increases occurred rather randomly for the roof tanks that were automatically filled regardless of the hour of the day (Figures 9a–c), and twice a day for the tanks that were programmed to be filled only in the morning and in the afternoon (Figures 9d–f). Since these increases were generally abrupt, this confirms the idea that it is possible to automatically distinguish tank-filling events. Nevertheless, it should be

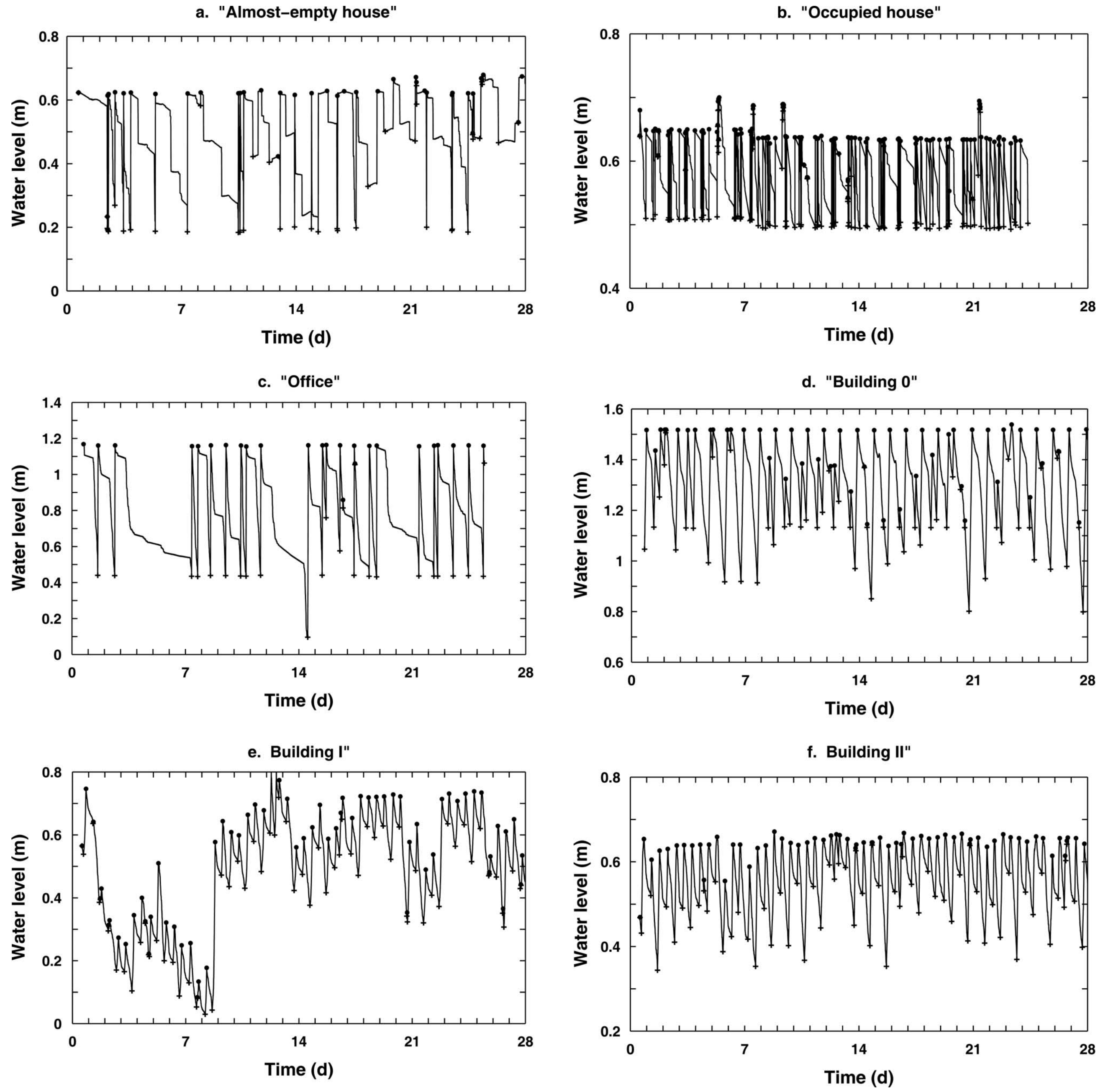

Figure 8. Raw data from the monitoring tests: water level in a roof tank as a function of time. The symbols represent local minima (circles) and maxima (crosses) that were automatically detected.

recognised that some increases in volume were not so abrupt in the case of the larger studied buildings (Figure 9d–f) and that this can create difficulties for estimating leaks using the proposed method (see related comments in Section 4.2.2).

Another clear trend is the slow and regular decrease in stored volume in the roof tanks, generally between ≈1:00 and 5:00. Since there were *a priori* no active persons in the studied buildings during these hours, this trend is consistent with the idea that it corresponds to indoor leaks. It is worth noting that a decrease of two different magnitudes is observed in the case of the "Occupied house" (Figure 9b; see related comments in Section 4.2.2).

Finally, it should be noted that incoherent data were obtained during the second half of the monitoring tests for the "Almost empty house". In fact, the water level registered in the roof tank increased slowly at times (Figure 8a, after day 12), when there were *a priori* no specific tank-filling events. This situation, which

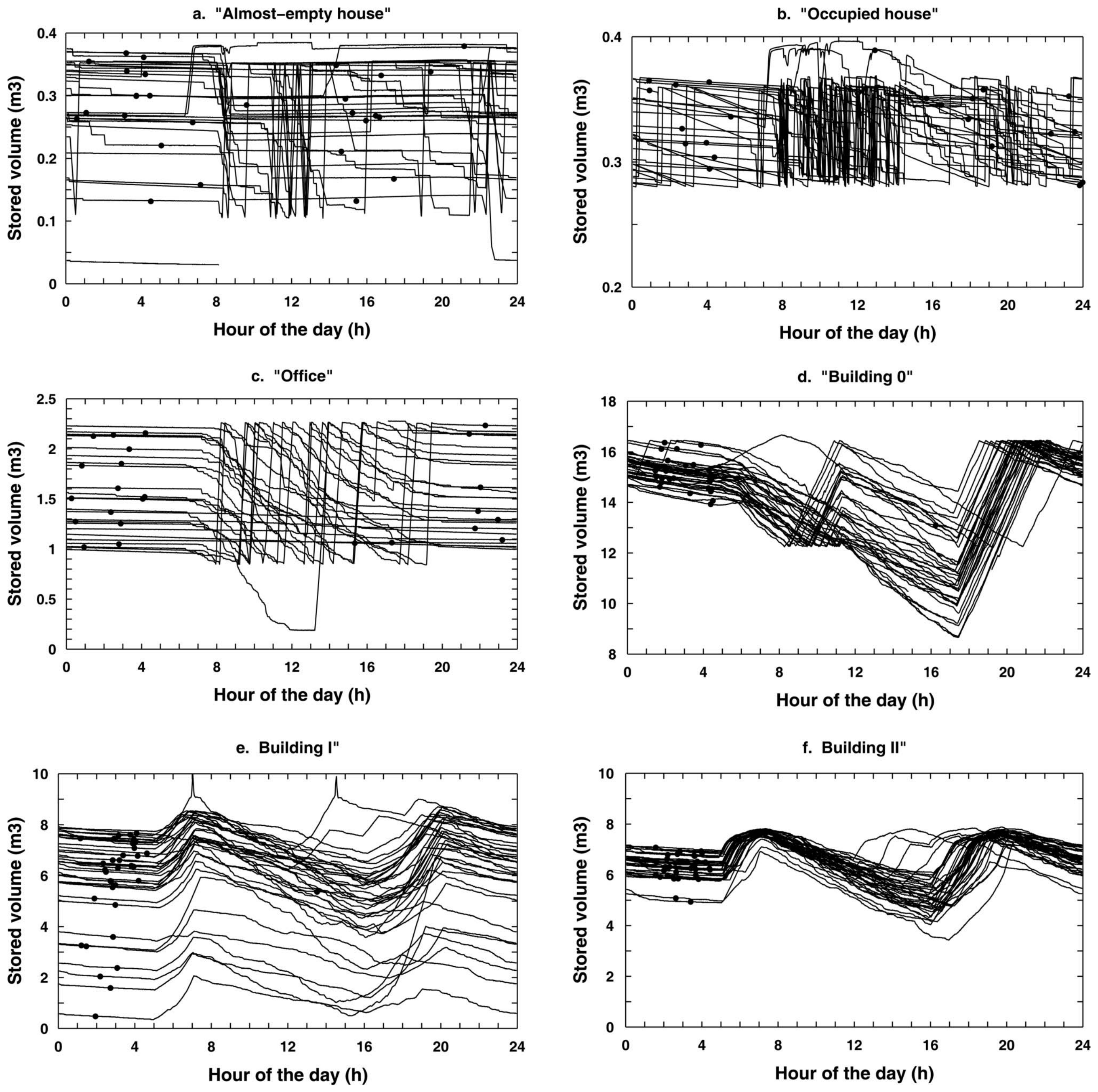

Figure 9. Raw data from the monitoring tests: stored volume in a roof tank as a function of hour of the day. The symbols represent the hours during which a leak value was automatically detected.

would be difficult to explain by a change in the geometry of the roof tank, was interpreted as a failure of the instrument used to monitor the water level because its battery was low ($\approx 30\%$ of its nominal capacity). Thus, it is likely that the instrument's sensitivity to air temperature had increased, although the instrument continued to respond to level changes and the data logger clock continued to be accurate. The consequences of this problem are presented below (Section 4.2.2).

4.2. *Analysis of leak estimations*

4.2.1. *Optimisation of the leak determination*

During the optimisation of the parameter $U(h)$ (Section 3.2.7), it was found that a $U(h)$ value between ≈ 2.5 mm and 3.5 mm was the best choice to estimate leaks in buildings with small or medium roof tanks (Figures 10a–c), while a value between ≈ 4.0 mm and 4.5 mm was best for buildings with a large roof tank (Figures 10d,f). In fact, for these $U(h)$

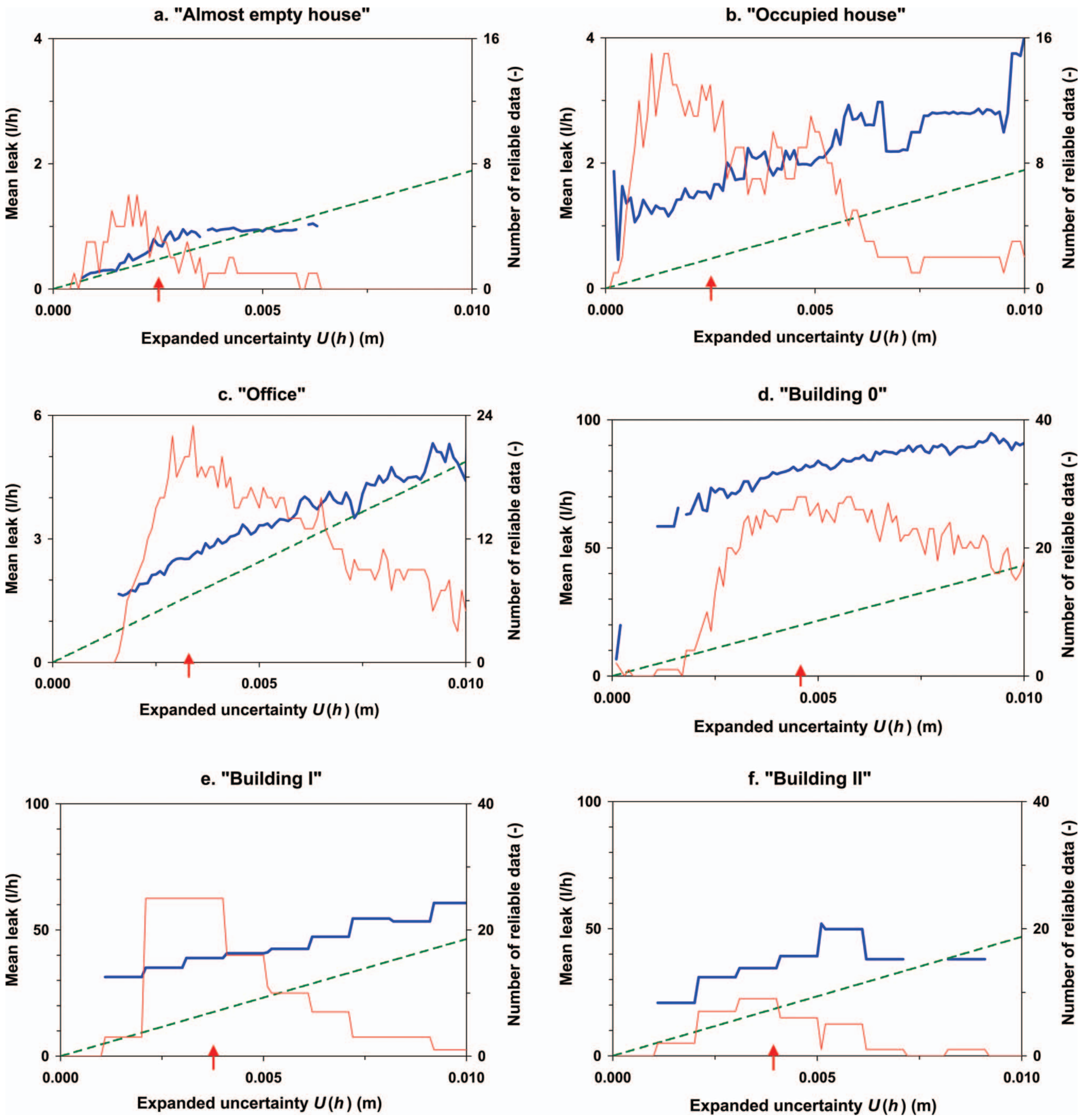

Figure 10. Optimisation of the leak determination: average of reliable leak estimations (thick line) and number of reliable estimations (thin line) as a function of the expanded uncertainty of water level data ($U(h)$). The arrow shows the $U(h)$ value used for the definitive calculations presented in this study (see Section 3.2.7). The dotted line shows the method's Detection Limit (see Section 3.2.6).

values, there were more days with a reliable leak estimation (N). This is consistent with the idea (Section 3.2.7) that for a large roof tank, it is more difficult to distinguish between a leak and intentional water consumption.

To estimate the method's Detection Limit (Section 3.2.6), it is necessary to know the maximum duration for which there is generally no intentional water consumption in a building and no tank-filling events. According to the visual analysis of the raw data (Figure 9), $\Delta t_{\max} = 6$ h was chosen for the "Almost empty house" and the "Occupied house", $\Delta t_{\max} = 7$ h for the "Office" and $\Delta t_{\max} = 5$ h for the three other buildings. In this case, the average leak estimation ($\varphi^{\bullet}$)

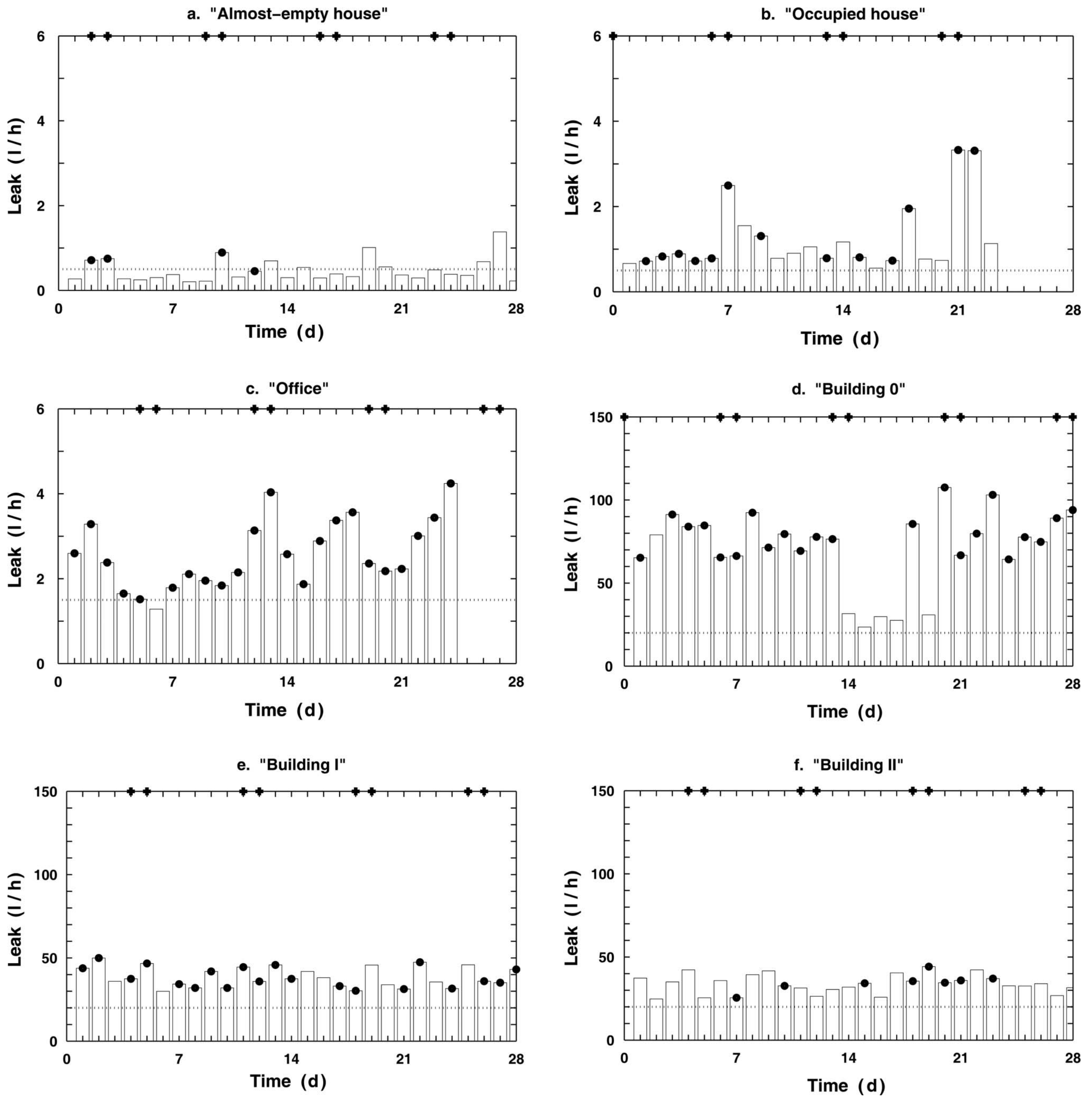

Figure 11. Leaks computed during each monitoring test in buildings with a roof tank. The bars represent leaks automatically determined by the proposed method. The circles show most reliable leak estimations and the dotted line shows the method's Detection Limit. The crosses at the top of each graph indicate weekends.

was quite close (less than double) to the Detection Limit in the cases of the "Almost empty house" (Figure 10a) and the "Office" (Figure 10c); in these cases, the average leak may have been overestimated (see Section 3.2.6).

As expected (Section 3.2.7), the optimisation also showed that the average leak estimation ($\varphi^\bullet$) tends to increase with the value of the parameter *U*(*h*). Nevertheless, this systematic error was acceptable for practical purposes; in fact, considering that the optimal value for *U*(*h*) is between 2.5 mm and 7.5 mm (Section 3.2.7), the $\varphi^\bullet$ estimate was consistent within $\pm 30\%$ for nearly all the studied buildings (Figures 10a and 10c–f). The only exception was the "Occupied house," where the $\varphi^\bullet$ estimation varied up to $\pm 60\%$ within the range of likely values for *U*(*h*). As mentioned below

(Section 4.2.2), this greater variability is due to the presence of leaks of two magnitudes; when choosing a value for $U(h)$ that is too large to determine leaks on a daily basis, the smaller ones are eliminated in the calculation of $\varphi^\bullet$.

4.2.2. Leak data

As expected, the leaks were usually estimated for the hours of the day between 1:00 and 5:00 (Figure 9). Nevertheless, some estimations were also obtained for other hours of the day in the case of the "Almost empty house" (Figure 9a) and the "Office" (Figure 9c). This is consistent with the fact that both buildings were not only unoccupied at night. It shows the versatility of the method for detecting leaks at any hour of the day.

The application of the method to the "Almost empty house" (Figure 11a) allows for only concluding that there were small indoor leaks (<0.7 l/h). Many unreliable leak estimations were automatically detected, because the leaks were very close to the method's Detection Limit (Section 4.2.1) and because the instrument used to monitor the water level was unfortunately unstable (Section 4.1.2).

In the case of the "Occupied house" (Figure 11b), leak estimates between 0.7 l/h and 3.5 l/h were obtained. Compared to the previous case, a larger leak was expected because there was not only one bathroom tap open in the house but rather three. Even though the smallest leak estimations are near the method's Detection Limit, they are realistic for a private house. In fact, indoor leaks in private houses are often <6 l/h (Arregui *et al.* 2006, Willis *et al.* 2009). In addition, the largest estimated leaks coincided with two weekends (days 7–9 and 21–22) during which an additional bathroom tap was open (which was only used when there were guests). At the end of the test, it was found that this bathroom indeed had a small leak, practically imperceptible to the eye. The results also indicate that a more significant leak must have been present on day 18 of the test, but the cause is unknown (perhaps a tap not properly closed).

In the case of the "Office" (Figure 11c), leak estimations larger than the previous two cases were obtained (between 1.5 l/h and 4 l/h). Nevertheless, in practice they are small for the size of the studied building, and they were near the method's Detection Limit (Section 4.2.1). It is worth noting that the building was equipped with devices (such as dry urinals) to save water and reduce domestic leakage.

In the case of "Building 0" (Figure 11d), leak estimations much larger than the previous three cases were obtained (between 60 l/h and 105 l/h). This is consistent with the idea that the magnitude of indoor leaks tends to increase with the size and age of a building (e.g., Willis *et al.* 2009). In addition, the leak estimations appear to be realistic because they are quite larger than the method's Detection Limit. Thanks to the linearity test (Section 3.2.5), questionable estimates were automatically detected on six occasions (e.g., days 14–17), which were due to slow tank-filling events (Figure 7b).

In the case of "Building I" (Figure 11e), leak estimations smaller than the previous case were obtained (between 35 l/h and 50 l/h). This is consistent with the fact that the building was newer (10 years-old as compared to 30 years) and therefore was probably in better condition. Finally, for "Building II" (Figure 11f), leak estimations were similar to those of the previous building. This is consistent with the fact that the two buildings are similar (in the same neighbourhood). Nevertheless, fewer reliable leak estimations were obtained, the cause of which is not well-known (some reasons may be slightly lower leak values, a slightly noisier instrument to monitor the water level from the roof tank, or some people in the building consuming water at night).

As expected, the leak magnitude was found to increase with the size and age of the six studied buildings (Table 2). In the worst case ("Building 0"), the estimated leak was as large as ≈ 40 l *per capita* per day, that is, more than 20% of the average drinking water consumption of a person in Mexico. Finally, little temporal variation in the leak data was found for each of the studied buildings (Coefficient of Variation $<30\%$), except in the case of a house with a damaged toilet ("Occupied house").

4.2.3. Difficulty to validate the proposed method

It would be desirable to validate the proposed method. Nevertheless, this is currently difficult to accomplish. One option would be to experimentally compare it with another method consisting of monitoring the flow from a roof tank outlet. Nonetheless, it is difficult to obtain a meter to monitor flows <10 l/h (Section 2.1.1). A second option would be to compare the method with another similar one that has been previously approved (EPA 2005), but the commercial instruments to monitor levels in tanks have not been developed for buildings with roof tanks, only for gasoline storage tanks (Section 2.1.3). A third option would be to experimentally simulate water leaks in a roof tank, though this would involve weeks of work using a sufficiently accurate peristaltic pump (Flora 2000). One final option would be to evaluate the method using numerical experiments based on a record of raw data ($h(t)$ relationship), in which some of the data are altered to simulate leaks of a known magnitude; nevertheless, this (Monte Carlo) approach

Table 2. Synthesis of leak estimates in buildings with a roof tank.

Case	a	b	c	d	e	f
Name	"Almost empty house "	"Occupied house"	"Office"	"Building 0"	"Building I"	"Building II"
Optimal value for expanded uncertainty of water level data ($U(h)$, mm; see Section 3.2.7)	2.5	2.5	3.4	4.5	4.0	4.0
Daily duration with no intentional water consumption (Δt_{max}, h; see Section 4.2.1)	6	6	7	5	5	5
Method's Detection Limit (DL, l/h; see Section 3.2.6)	0.5	0.5	1.5	20	20	20
Obtained number of reliable leak estimates (N; see Section 3.2.7)	4(a)	13	23	28	25	9
Leak in the building: average ($\varphi^{\bullet}$, l/h) and standard deviation of reliable data	<0.7(b) (0.2)	1.4 (1.0)	<2.6(b) (0.7)	80 (11)	39 (6)	35 (5)
Weighted leak (liters *per capita* per day)	<17	11	<1	40	22	17

(a) Many leak estimations were eliminated because the leak was small and the instrument to monitor the water depth was failing (low battery).
(b) Value near the method's Detection Limit. In this case, the leak may have been over-estimated (see Section 3.2.6).

involves performing a large number of calculations with raw data previously collected from various buildings (Flora 2000).

Although it has not yet been possible to validate the proposed method, it has provided data consistent with what was expected (Section 4.2.2). In addition, it includes various options to check the goodness of the results. In particular, the manner in which the $U(h)$ parameter is optimised (Section 3.2.7) also serves as a sensitivity analysis; in this regard, the results indicate that for the studied buildings, leak estimations ($\varphi^{\bullet}$) were consistent within a range of $\pm 30\%$ (Section 4.2.1).

5. Conclusion

Buildings with intermittent service and a roof tank are probably not the best technical solution in terms of water supply, but they are common in many countries with few water resources. A method has been proposed to determine the leak inside these buildings. This study explains how to set up the method and shows its application to six cases. As expected, the leak magnitude was found to increase with the size and age of buildings. In the worst case, a leak of 40 l *per capita* per day was detected. Although it is difficult to validate, the method includes various options to check the goodness of the results. In practice, it can estimate (within $\pm$ 30%) a leak as small as 1 l/h when the roof tank is small (cross-sectional area ≈ 0.5 m^2), 4 l/h when the tank is medium (area ≈ 2 m^2) and 20 l/h when the tank is large (area ≈ 10 m^2).

The proposed method is simple to set up experimentally. However, due to its cost (equal to leaving an instrument of \$ 1500 USD in a roof tank for several days) and the complexity of data processing (a code for that purpose is available upon request), it should be seen as a new tool for researchers who investigate water losses in areas with intermittent supply. The method can be applied to different kinds of buildings (private buildings, residential houses, small factories, laundromats, hotels, restaurants, bars). Finally, it should be noted that under certain conditions (a roof tank filling quickly and not always during the same hours of the day), it can be extended to determine not only leaks but also a water consumption pattern (Tamari and Alcocer-Yamanaka 2012).

Acknowledgments

Any reference to a commercial model and its manufacturer is for product identification purposes only, and does not represent an endorsement of the product. Thank you to Prof. Bruce E. Hansen for making the software available for conducting the statistical tests described in his 1992 publication (www.ssc.wisc.edu/~bhansen/ [Accessed 26 October 2011]), and to Pilar Guerrero for her great help.

References

Andey, S.P. and Kelkar, P.S., 2009. Influence of intermittent and continuous modes of water supply on domestic water consumption. *Water Resources Management*, 23 (12), 2555–2566.

Arregui, F., Cabrera Jr., E., and Cobacho, R., 2006. *Integrated water meter management*. London: IWA.

Cobacho, R., *et al.*, 2008. Private water storage tanks: evaluating their inefficiencies. *Water Practice & Technology*, 3 (1), 8.

Criminisi, A., *et al.*, 2009. Evaluation of the apparent losses caused by water meter under-registration in intermittent water supply. Water Science and Technology, 60 (9), 2373–2382.

EPA, 2005. *Straight talk on tanks: leak detection methods for petroleum underground storage tanks and piping (EPA-510-B-05-001)* [online]. US Environmental Protection Agency (USEPA). Available from: www.epa.gov/oust/pubs/straight.htm [Accessed 26 October 2011].

Flora, J.D., 2000. *Evaluation protocol for continuous in-tank leak detection systems* [online]. Kansas City (MO): Midwest Research Institute (MRI). Available from: www.nwglde.org/protocols.html [Accessed 26 October 2011].

Grafton, R.Q., *et al*., 2009. *Residential water consumption: a cross country analysis* [online]. EERH Research Report No. 23, Crawford School of Economics and Government, Australian National University, Canberra, Australia. Available from: www.crawford.anu.edu.au/research_units/eerh/publications/ [Accessed 26 October 2011]

Hansen, B.E. 1992. Testing for parameter instability in linear models. *Journal of Policy Modeling*, 14 (4), 517–533.

JCGM, 2008. *Evaluation of measurement data—Guide to the expression of uncertainty in measurement (JCGM 100:2008)*. Paris: Working Group 1 of the Joint Committee for Guides in Metrology (JCGM/WG 1). Available from: www.bipm.org/fr/publications/guides/ [Accessed 26 December 2011]

Lambert, A.O., 2002. International report: water losses management and techniques. *Water Science and Technology: Water Supply*, 2 (4), 1–20.

Lee, E.J. and Schwab, K.J., 2005. Deficiencies in drinking water distribution systems in developing countries. *J. Water and Health*, 3 (2), 109–127.

NWGLDE, 2007. What is CITLDs leak detection all about [online]. *LUSTLine*, 56, 18. Available from: www.nwglde.org [Accessed 26 October 2011]

Otaki, Y., *et al*., 2008. Micro-components survey of residential indoor water consumption in Chiang Mai. Drink. *Water Eng. Sci*., 1, 17–25.

OIML, 2008. *Automatic level gauges for measuring the level of liquid in stationary storage tanks (OIML R-85: 2008(E))*. Paris: Organisation Internationale de Métrologie Légale (OIML).

Puust, R., *et al*., 2010. A review of methods for leakage management in pipe networks. *Urban Water Journal*, 17 (1), 25–45.

Tamari, S., Laporte-Vergnes, A., and Salgado, G., 2010. A simple benchmark to test handheld laser distance meters [online]. *In*: C.A. Galván-Hernández, ed. *Proc. "Simposio de Metrología 2010"*, Centro Nacional de Metrología (CENAM), Queretaro, México, 27–29 October 2010, paper C17. Available from: www.cenam.mx/simposio 2010/ [Accessed 26 October 2011] [in Spanish].

Tamari, S. and Aguilar-Chávez, A., 2010. Testing submersible pressure transducers to monitor water level in tanks. *Tecnología y Ciencias del Agua*, 1 (3), 71–88. [in Spanish].

Tamari, S. and Alcocer-Yamanaka, V.H., 2012. A method to determine water consumption in buildings with a roof tank - 1. Data acquisition. Water SA (Online) [submitted].

Trifunovic, N., 2006. *Introduction to urban water distribution*. Leiden, The Netherlands: Taylor & Francis/Balkema.

Willis, R., Stewart, R., Panuwatwanich, K., Capati, B., and Giurco, D., 2009. Gold Coast domestic water end use study. *Water: Journal of the Australian Water Association*, 36 (6), 91–97.

```
' ***********************************************************
' Program:      LIDAR_S200.CR1
'
' Objective:    Acquire data from LIDAR ("Laser Technology")
'               COM1:  TRU  ("True Sense S200")
'
' Compilation:  CRBasic Editor vers. 3.1
' Datalogger:   CR1000 Series Datalogger   (source: *.CR1)
'
' Author:       Serge Tamari + Younes Rifad
' ***********************************************************
'


' DECLARE VARIABLES
' *****************
Dim RT(9)      'Array of current date


' PARAMETERS
' ==========

' Time step (min)
Const Time_Step =  10      ' <<< TIMING >>>


' Number PI
Const PI = 3.141592654


' INTERNAL VARIABLES
' ==================

' Flag (0 = not ready, 1 = Ok)
Public istatus

' Iteration number
Public iloop

' Dummy numbers
Public Dummy
Public Dummy_List(8)

' Dummy texts
Public Dummy_Text     As String * 200
Public Dummy_Texti    As String * 200
Public Text_List(8)   As String * 200

' Data received from any COM (text)
Public Com_Text       As String * 200

' Null variables
Public Null_Text      As String * 200
Public Null_List(8)


' ERROR MESSAGES
' ==============

' File handle (number)
Public Filou

' Warning Text(text)
Public Warning       As String * 200


' TIME DATA
' =========

' Time to collect data within each iteration (sec)
Public Time_delay
```

```
' DATALOGGER DATA
' ===============

' Datalogger temperature (C)
Public Temperature

' Datalogger input voltage (V)
Public Voltage


' LIDAR DATA - TRU
' ================

' Configuration code
Public TRU_kmode

' Distance - Mode "First"
Public TRU_Reading1  As String * 200       ' Sent by COM
Public TRU_Data1(8)                        ' Data array
Alias  TRU_Data1(1)  = TRU_D1              ' Distance (m)
Alias  TRU_Data1(2)  = TRU_K1              ' Error code (0 = Ok)
Alias  TRU_Data1(3)  = TRU_I1              ' Signal strength [a]
Alias  TRU_Data1(4)  = TRU_J1              ' Signal strength [b]

' Distance - Mode "Strongest"
Public TRU_Reading2  As String * 200       ' Sent by COM
Public TRU_Data2(8)                        ' Data array
Alias  TRU_Data2(1)  = TRU_D2              ' Distance (m)
Alias  TRU_Data2(2)  = TRU_K2              ' Error code (0 = Ok)
Alias  TRU_Data2(3)  = TRU_I2              ' Signal strength [a]
Alias  TRU_Data2(4)  = TRU_J2              ' Signal strength [b]

' Distance - Mode "Last"
Public TRU_Reading3  As String * 200       ' Sent by COM
Public TRU_Data3(8)                        ' Data array
Alias  TRU_Data3(1)  = TRU_D3              ' Distance (m)
Alias  TRU_Data3(2)  = TRU_K3              ' Error code (0 = Ok)
Alias  TRU_Data3(3)  = TRU_I3              ' Signal strength [a]
Alias  TRU_Data3(4)  = TRU_J3              ' Signal strength [b]

' Temperature of Lidar (C)
Public TRU_Temperature


' DEFINE THE OUTPUT DATA FILE
' ==========================

' Define the output data file
DataTable (Data_File,True,-1)             ' Use real datalogger clock ?

  DataInterval (0, Time_Step ,3,10)       ' Time step to store data  <<< TIMING >>>

  Sample (1 , Voltage     , FP2 )         ' Battery
  Sample (1 , Temperature , FP2 )         ' Datalogger temperature

  Sample (1 , TRU_D1      , IEEE4 )       ' Data from Laser "TRU"
  Sample (1 , TRU_K1      , UINT2 )
  Sample (1 , TRU_I1      , UINT2 )
  Sample (1 , TRU_J1      , UINT2 )
  Sample (1 , TRU_D2      , IEEE4 )
  Sample (1 , TRU_K2      , UINT2 )
  Sample (1 , TRU_I2      , UINT2 )
  Sample (1 , TRU_J2      , UINT2 )
  Sample (1 , TRU_D3      , IEEE4 )
  Sample (1 , TRU_K3      , UINT2 )
  Sample (1 , TRU_I3      , UINT2 )
  Sample (1 , TRU_J3      , UINT2 )

  Sample (1 , TRU_Temperature , FP2 )    ' Temperature of Laser "TRU"
```

```
EndTable


' Compilation option
SequentialMode

'------------------------------------------------------------



'------------------------------------------------------------
' SUBROUTINE Warn
' ***************
'
' Objective:    Write a message into an ASCII file
' Input:        Warning (text)
' Version:      July 6, 2011
' Author:       Serge Tamari + Younes Rifad

Sub Warn

  Filou = FileOpen( "CPU:Error_File" , "a" , -1 )

  FileWrite ( Filou , Warning , 0 )
  FileWrite ( Filou , CHR(13) , 0 )

  FileClose ( Filou )


EndSub
'------------------------------------------------------------



'------------------------------------------------------------
' SUBROUTINE Ini_TRU
' ******************
'
' Objective:    Initialize the Laser TRU
' Input:        Mode of configuration of the laser
' Output:       Data list
' Version:      July 13, 2011
' Author:       Younes Rifad + Serge Tamari


Sub Ini_TRU


  ' Banner for the error file
  ' ========================

  Warning = "  "
  Call Warn
  Warning = "  INITIALIZE TRU...       "
  Call Warn
  Warning = " ----------------------- "
  Call Warn
  Warning = "  "
  Call Warn


  ' STOP OPERATION BEFORE CONFIGURATION
  ' ==================================

  ' Select STOP command
  SerialOut (Com1,  "$ST"          ,"",0,4)
  SerialOut (Com1,  CHR (13)       ,"",0,20)

  ' Wait for answer
  SerialOut (Com1,"","",10,100)
```

```
' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 3 ) = "$OK" ) Then
  Warning = " TRU :  $ST --- Pass "
Else
  Warning = " TRU :  $ST --- Fail !!! "
EndIf
Call Warn
SerialFlush(Com1)

' User password ("$OK" or "$ER")   [Default = <admin>]
SerialOut (Com1,  "$PW,admin"    ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$PW"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 3 ) = "$OK" ) Then
  Warning = " TRU :  $PW --- Pass      "
Else
  Warning = " TRU :  $PW --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Remote trigger (0 = Off, so that "$GO" and "$ST" can be used)   + NEED PASSWORD +
SerialOut (Com1,  "$PW,admin"    ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)
SerialFlush(Com1)
SerialOut (Com1,  "$TG,0"        ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$TG,0" ) Then
  Warning = " TRU :  $TG --- Pass "
Else
  Warning = " TRU :  $TG --- Fail !!! "
EndIf
Call Warn
SerialFlush(Com1)


' OPTIONS FOR OPERATION
' ====================

' Manual start (0 = Manual & needs "$GO" ; 2 = Automatic)
SerialOut (Com1,  "$MA,0"         ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$MA"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)
```

```
' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$MA,0" ) Then
  Warning = " TRU :  $MA --- Pass      "
Else
  Warning = " TRU :  $MA --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Measurement unit (0 or "M" = Meters)         + NOT AS IN USER'S MANUAL ! +
SerialOut (Com1,  "$MU,0"         ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$MU"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 7 ) = "$MU,M,2" ) Then
  Warning = " TRU :  $MU --- Pass      "
Else
  Warning = " TRU :  $MU --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Time since power on, in seconds (0 = disabled, 2 = enabled)
' <<< Not used, but necessary during data extraction >>>
SerialOut (Com1,  "$DT,2"         ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$DT"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$DT,2" ) Then
  Warning = " TRU :  $DT --- Pass      "
Else
  Warning = " TRU :  $DT --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Error code format (0 = code only, 4 = code with mnemonic)
SerialOut (Com1,  "$DE,0"         ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$DE"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)
```

```
' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$DE,0" ) Then
  Warning = " TRU :  $DE --- Pass      "
Else
  Warning = " TRU :  $DE --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)


' OPTIONS TO ACQUIRE DATA
' =======================

' Select NUMBER OF PULSE PER MEASUREMENT     (<16> = 12 data per second !)
SerialOut (Com1,  "$OP,16"       ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$OP"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 6 ) = "$OP,16" ) Then
  Warning = " TRU :  $OP --- Pass      "
Else
  Warning = " TRU :  $OP --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Warm up period (0 = Off, 1 ... 99 = Discarded measurements [6 = Default])
SerialOut (Com1,  "$WU,6"        ,"",0,4)
SerialOut (Com1,  CHR (13)        ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$WU"       ,"",0,4)
SerialOut (Com1,  CHR (13)    ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$WU,6" ) Then
  Warning = " TRU :  $WU --- Pass      "
Else
  Warning = " TRU :  $WU --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

' Target mode  ... This is only a check here       + NEED PASSWORD +

' First Target     : TRU_kmode = 2  Answer: "$DM,2"  Data: "$DM,F,..."
' Strongest Target : TRU_kmode = 3  Answer: "$DM,3"  Data: "$DM,S,..."
' Last Target      : TRU_kmode = 4  Answer: "$DM,4"  Data: "$DM,L,..."

' NOTE: option "TRU_kmode = 7" does not allow to get "Signal Strength" !


SerialOut (Com1,  "$PW,admin"    ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)
SerialFlush(Com1)
SerialOut (Com1,  "$DM,2"        ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$DM"      ,"",0,4)
SerialOut (Com1,  CHR (13)   ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$DM,2" ) Then
  Warning = " TRU :  $DM,2 --- Pass      "
Else
  Warning = " TRU :  $DM,2 --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

SerialOut (Com1,  "$PW,admin"    ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)
SerialFlush(Com1)
SerialOut (Com1,  "$DM,3"        ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$DM"      ,"",0,4)
SerialOut (Com1,  CHR (13)   ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$DM,3" ) Then
  Warning = " TRU :  $DM,3 --- Pass      "
Else
  Warning = " TRU :  $DM,3 --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)

SerialOut (Com1,  "$PW,admin"    ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)
SerialFlush(Com1)
SerialOut (Com1,  "$DM,4"        ,"",0,4)
SerialOut (Com1,  CHR (13)       ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$DM"      ,"",0,4)
SerialOut (Com1,  CHR (13)   ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$DM,4" ) Then
  Warning = " TRU :  $DM,4 --- Pass      "
Else
  Warning = " TRU :  $DM,4 --- Fail !!!  "
EndIf
```

```
Call Warn
SerialFlush(Com1)

' Measurement mode (0 = Standard range, up to 750 m)
SerialOut (Com1,  "$MM,0"       ,"",0,4)
SerialOut (Com1,  CHR (13)      ,"",0,20)

' Validate and clean
SerialFlush(Com1)
SerialOut (Com1,  "$MM"     ,"",0,4)
SerialOut (Com1,  CHR (13)  ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 5 ) = "$MM,0" ) Then
  Warning = " TRU :  $MM --- Pass      "
Else
  Warning = " TRU :  $MM --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)


' SAVE CONFIGURATION
' ==================

' Save user settings to flash memory
SerialOut (Com1,  "$SU"         ,"",0,4)
SerialOut (Com1,  CHR (13)      ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 3 ) = "$OK" ) Then
  Warning = " TRU :  $SU --- Pass      "
Else
  Warning = " TRU :  $SU --- Fail !!!  "
EndIf
Call Warn
SerialFlush(Com1)


' Read Instrument status (flag, error, password)
' Flag:     0 = not firing,  1 = firing
' Error:    0 = Ok        , >0 = error code
' Password: 0 = Not active, 1 or 2 = Active   + not tested +

SerialOut (Com1,  "$IS"        ,"",0,4)
SerialOut (Com1,  CHR (13)     ,"",0,20)

' Wait for answer
SerialOut (Com1,"","",10,100)

' Check answer
Dummy_Text = Null_Text
SerialIn ( Dummy_Text ,Com1,100,0,1000)
If ( Mid( Dummy_Text , 1 , 7 ) = "$IS,0,0" ) Then
  Warning = " TRU :  $IS --- Pass "
Else
  Warning = " TRU :  $IS --- Fail !!! "
EndIf
Call Warn
SerialFlush(Com1)
```

```
EndSub
'-------------------------------------------------------------


'-------------------------------------------------------------
' SUBROUTINE Lec_TRU
' ******************
'
' Objective:    Read data from the Laser TRU SENSE
' Input:        Mode of configuration of the Laser
' Output:       Data list
' Version:      July 14, 2011
' Author:       Serge Tamari + Younes Rifad


Sub Lec_TRU

  ' Select the mode (command "DM,TRU_kmode")
  ' ========================================

  ' First Target     : TRU_kmode = 2  Answer: "$DM,2"  Data: "$DM,F,..."
  ' Strongest Target : TRU_kmode = 3  Answer: "$DM,3"  Data: "$DM,S,..."
  ' Last Target      : TRU_kmode = 4  Answer: "$DM,4"  Data: "$DM,L,..."


  ' Try a few times +++
  iloop   = 0
  istatus = 0
  While ( iloop < 8 ) AND ( istatus = 0 )

    ' Loop increment
    iloop = iloop + 1

    ' PASSWORD REQUIRED EACH TIME !
    SerialOut (Com1,  "$PW,admin" + CHR(13)       ,"",0,200)
    SerialOut (Com1,"","",10,100)
    SerialFlush (Com1)

    ' Request
    If ( TRU_kmode = 2 ) Then

      ' Mode "Averaging"
      SerialOut (Com1,"$DM,2" + CHR(13) ,"",0,200)
      SerialOut (Com1,"","",10,100)

    ElseIf ( TRU_kmode = 3) Then

      ' Mode "Binning"
      SerialOut (Com1,"$DM,3" + CHR(13) ,"",0,200)
      SerialOut (Com1,"","",10,100)

    ElseIf ( TRU_kmode = 4) Then

      ' Mode "Last Target"
      SerialOut (Com1,"$DM,4" + CHR(13) ,"",0,200)
      SerialOut (Com1,"","",10,100)

    Else

      ' No other available mode     +++ FATAL ! Stop the program... +++
      Exit

    EndIf

    ' Validate the mode selection  +++ Basic +++
    SerialFlush (Com1)
    SerialOut (Com1,  "$DM" + CHR(13)  ,"",0,200)

    ' Check the answer
    SerialOut (Com1,"","",10,100)
    Dummy_Text = Null_Text
    SerialIn ( Dummy_Text ,Com1,100,0,1000)
    SerialFlush (Com1)

    Dummy_Texti = Mid(Dummy_Text , 1 , 3 )     ' Extract some characters
    SplitStr ( Dummy , Dummy_Text ,",",1,0)    ' Extract a number

    If ( Dummy_Texti <> "$DM" ) OR ( Dummy <> TRU_kmode ) Then
      istatus = 0
    Else
      istatus = 1
    EndIf

  Wend



  ' Check the status
  ' ================
  If ( istatus = 0) Then
    Warning = " Read TRU :  $DM --- Fail !!! "
    Call Warn
  EndIf


  ' Save user settings to flash memory
  ' ==================================

  SerialOut (Com1,  "$SU" + CHR(13)      ,"",0,200)

  ' Check answer
  SerialOut (Com1,"","",10,100)
  Dummy_Text = Null_Text
  SerialIn ( Dummy_Text ,Com1,100,0,1000)
  SerialFlush (Com1)
  If ( Mid( Dummy_Text , 1 , 3 ) = "$OK" ) Then
    istatus = 1
  Else
    istatus = 0
    Warning = " Read TRU :  $SU --- Fail !!! "
    Call Warn
  EndIf


  ' Read the data
  ' =============

  ' Read if Ok...
  If ( istatus = 1 ) Then

    ' Ask for data
    ' ============

    ' Try a few times ++++  [2012.05.09 >>> try 16 times instead of 8]
    iloop   = 0
    istatus = 0
    While ( iloop < 16 ) AND ( istatus = 0 )

      ' Loop increment
      iloop = iloop + 1

      ' Clean the buffer
      SerialFlush (Com1)

      ' Data request (command "$GO,1")
      SerialOut (Com1,"$GO,1" + CHR(13) ,"",0,200)

      ' Read the answer    = A SINGLE LINE !!!
      SerialOut (Com1,"","",10,100)
      Dummy_Text = Null_Text
      SerialIn ( Dummy_Text ,Com1,100,0,1000)
      SerialFlush (Com1)
```

```
      ' Check the FIRST PART of answer ("$OK")
      If ( Mid( Dummy_Text, 1 , 3 ) = "$OK" ) Then
        istatus = 1
      Else
        istatus = 0
        Warning = " Read TRU :  $GO --- Fail !!! "
        Call Warn
        Com_Text = Null_Text
      EndIf

      ' Extract the SECOND PART of answer ("$DM,...")
      Com_Text = Mid (Dummy_Text , 11 , Len(Dummy_Text)-10 )

      ' Check the answer (Banner only = "$DM,x")
      If     (TRU_kmode = 2) AND (Mid(Com_Text,1,5) = "$DM,F") Then
        istatus = 1
      ElseIf (TRU_kmode = 3) AND (Mid(Com_Text,1,5) = "$DM,S") Then
        istatus = 1
      ElseIf (TRU_kmode = 4) AND (Mid(Com_Text,1,5) = "$DM,L") Then
        istatus = 1
      Else
        istatus = 0
        Warning = " Read TRU :  $DM,x --- Fail !!! "
        Com_Text = Null_Text
        Call Warn
      EndIf

    Wend


    ' Could not get data...
  Else

    ' Cannot get data
    Com_Text = Null_Text

    ' "Warning" message
    Warning = " Read TRU : Cannot get data..."
    ''' [may create a very large file !!! >>>  Call Warn

  EndIf


EndSub
'-------------------------------------------------------------



'-------------------------------------------------------------
' MAIN PROGRAM
' ************

BeginProg


  ' INITIALIZATION
  ' ==============

  ' Banner for the error file
  Warning = "  "
  Call Warn
  Warning = " ----------------------- "
  Call Warn
  Warning = "  LIDAR  - ERROR FILE     "
  Call Warn
  Warning = " ----------------------- "
  Call Warn


  ' Open and configure "COM1"
```

```
SerialOpen (Com1,115200,19,100,1000)

Call Ini_TRU


' ITERATION LOOP
' ==============

' Time step for iterations       ' <<< TIMING >>>
Scan( Time_Step ,3, 1 , 0 )

  ' Initialize the delay (sec) during an iterartion
  Time_delay = Timer (1,sec,2)

  ' Initialize the "warning" messages
  Warning = "Ok..."

  ' Get the battery voltage
  Battery ( Voltage )

  ' Get the datalogger temperature
  PanelTemp ( Temperature , _60Hz )


  ' CONFIGURE AND READ DATA OF LIDAR (TRU SENSE S200)
  ' ================================================

  ' First Target    : TRU_kmode = 2  Answer: "$DM,2"  Data: "$DM,F,..."
  ' Strongest Target: TRU_kmode = 3  Answer: "$DM,3"  Data: "$DM,S,..."
  ' Last Target     : TRU_kmode = 4  Answer: "$DM,4"  Data: "$DM,L,..."


  ' SELECT THE 'FIRST TARGET' MODE
  ' ------------------------------

  ' Configure Laser and read data
  TRU_kmode = 2
  Com_Text = Null_Text
  Call Lec_TRU
  TRU_Reading1 = Com_Text

  ' Extract the data (SEVEN PIECES OF TEXT)
  TRU_Data1 = Null_List
  Text_List = Null_Text
  If ( istatus = 1)
    SplitStr ( Text_List , TRU_Reading1 , "," ,7,4)
    TRU_Data1(1) = Text_List(2)                          ' Distance (m)
    TRU_Data1(2) = Text_List(4)                          ' Error code
    TRU_Data1(3) = Mid( Text_List(5) , 1 , 1 )           ' Signal [a]
    TRU_Data1(4) = Mid(Text_List(5),3,Len(Text_List(5))) ' Signal [b]
  Else
    TRU_Data1(1) = NaN
    TRU_Data1(2) = NaN
    TRU_Data1(3) = NaN
    TRU_Data1(4) = NaN
  EndIf


  ' SELECT THE 'STRONGEST TARGET' MODE
  ' ----------------------------------

  ' Configure Laser and read data
  TRU_kmode = 3
  Com_Text = Null_Text
  Call Lec_TRU
  TRU_Reading2 = Com_Text

  ' Extract the data (SEVEN PIECES OF TEXT)
  TRU_Data2 = Null_List
  Text_List = Null_List
```

```
If ( istatus = 1)
  SplitStr ( Text_List , TRU_Reading2 , "," ,7,4)
  TRU_Data2(1) = Text_List(2)                          ' Distance (m)
  TRU_Data2(2) = Text_List(4)                          ' Error code
  TRU_Data2(3) = Mid ( Text_List(5) , 1 , 1 )          ' Signal [a]
  TRU_Data2(4) = Mid(Text_List(5),3,Len(Text_List(5))) ' Signal [b]
Else
  TRU_Data2(1) = NaN
  TRU_Data2(2) = NaN
  TRU_Data2(3) = NaN
  TRU_Data2(4) = NaN
EndIf


' SELECT THE 'LAST TARGET' MODE
' ----------------------------

' Configure Laser and read data
TRU_kmode = 4
Com_Text = Null_Text
Call Lec_TRU
TRU_Reading3 = Com_Text

' Extract the data (SEVEN PIECES OF TEXT)
TRU_Data3 = Null_List
Text_List = Null_List
If ( istatus = 1)
  SplitStr ( Text_List , TRU_Reading3 , "," ,7,4)
  TRU_Data3(1) = Text_List(2)                          ' Distance (m)
  TRU_Data3(2) = Text_List(4)                           ' Error code
  TRU_Data3(3) = Mid ( Text_List(5) , 1 , 1 )           ' Signal [a]
  TRU_Data3(4) = Mid(Text_List(5),3,Len(Text_List(5)))  ' Signal [b]
Else
  TRU_Data3(1) = NaN
  TRU_Data3(2) = NaN
  TRU_Data3(3) = NaN
  TRU_Data3(4) = NaN
EndIf


' READ THE INSTRUMENT TEMPERATURE (C)
' ----------------------------------

' Send command
SerialFlush (Com1)
SerialOut (Com1,  "$OZ" + CHR(13)  ,"",0,200)
SerialOut (Com1,"","",10,100)

' Check the answer
Dummy_Text = Null_Text
Dummy      = 0
SerialIn ( Dummy_Text ,Com1,100,0,1000)
SerialFlush (Com1)

Dummy_Texti = Left( Dummy_Text , 3 )        ' Extract some characters
SplitStr ( Dummy , Dummy_Text ,",",1,0)       ' Extract a number

If ( Dummy_Texti <> "$OZ" ) Then
  TRU_Temperature = NaN
Else
  TRU_Temperature = Dummy
EndIf


' END OF LOOP
' ===========

' Get the delay (sec) during an iteration
Time_delay = Timer (1,sec,4)

Time_delay = Time_delay / 60             ' Convert to min.
```

```
    If ( Time_delay > Time_Step ) Then
      Warning = " Warning: Time_delay > Time_Step !!!"
      Call Warn
    EndIf


    ' Write data in the output file
    CallTable Data_File


    ' Cellular modem (turn off from 4:59 PM to 10:59 AM)
    RealTime (RT())
    If RT(4) >= 10 AND RT(4)<= 16 Then
      SW12 (1 )
    Else
      SW12 (0)
    EndIf


  ' Go back
  NextScan


' END OF MAIN
EndProg
'-------------------------------------------------------------
```

```
% ---------------------------------------------------------------
% Funcion:    klik.m
%
% Objectivo:  Programa que lee los datos de monitoreo en tinaco
%             y los procesa para analizar las fugas (basado en
%             el programa 'consumo.m').
%
%             Simula un monitoreo 'manual' (entre dos horas del dia)
%
% Hipotesis:  El area del tinaco es constante
%             El tinaco se llena solo cuando el tirante aumenta
%             Se detectan los tirantes mini-maxi de llenado
%
%             Nota: el programa utiliza 3 parametros para detectar
%                   automaticamente los tirantes mini-maxi, estos
%                   parametros son: 'tolhini', 'tolhfin', 'klis'.
%
%             Nota: 'klis = -1'significa que se define manualmente
%                   la lista de los minima (imin) y maxima (imax)...
%
%             Nota: el programa utiliza 1 parametro ('tolh') para
%                   guardar los datos que corresponden a cambios de
%                   tirante 'significativos'.
%
% Datos:      Archivo de texto con 2 columnas de datos
%                   Columna 1 = Tiempo acumulado (dias)
%                   Columna 2 = Tirante medido   (m)
%
%             Nota: normalmente, el tiempo empieza desde cero,
%                   y se indica la hora de inicio en el programa.
%
%             Nota: en el programa, se indican los fines de semana
%                   (sabado - domingo) como días contados desde el
%                   primer día de monitoreo (es decir, desde cero)
%
%             Nota: normalmente, se mide el tirante con respecto
%                   al fondo del tinaco.
%
%             Nota: las unidades que se manejan en el programa son
%                   horas, metros y litros.
%
% ---------------------------------------------------------------


% Lee la tabla de datos crudos y define los parametros
% ==================================================

  %- Lee los parametros y los datos de la prueba
  %  La tabla de datos crudos es regular
  %  Columna 1 = Tiempo acumulado (dias)
  %  Columna 2 = Tirante en el tinaco (m)


   disp(' ')
   disp(' ')
   disp(' ')
   disp(' ')
   disp(' ************************************************')
   disp(' *              FUGAS DE AGUA                   *')
   disp(' *          EN CASAS CON TINACO                 *')
   disp(' *                             (c) Serge 2012 *')
   disp(' ************************************************')

   disp(' ')
   disp([' Fecha de los calculos : ' date ])

   disp(' ')
   disp(' [0] Monitoreo en casa casi-vacia       ')
   disp(' [1] Monitoreo en casa privada          ')
   disp(' [2] Monitoreo en edificio del gobierno ')
   disp(' [3] Monitoreo en edificio privado      ')
   disp(' [4] Monitoreo en edificio <A>  VALIDA  ')
   disp(' [5] Monitoreo en edificio <B>          ')
   disp(' [6] Monitoreo en edificio <A>          ')
   disp(' [7] Monitoreo en edificio <B>  VALIDA  ')


   nrep = input(' No prueba deseada -> ' );
   disp(' ')


   if (nrep == 0)

   % ---------------------------------------------------------------------
     disp(' Lugar    : Tinaco de casa particular [Serge Tamari] ')
     disp(' Tinaco   : Cilíndrico - capacidad de hasta 450 L    ')
     disp(' Inicio   : Jueves 22 de abril 2010 a las 17:00      ')
     disp(' Final    : Viernes 21 de mayo 2010 a las  8:07      ')
     disp(' Duracion : -----------   ')
     disp(' Datos    : -----------   ')
     disp(' Sensor   : Keller # 0022 (rango 5 mca) [Bateria baja (30 %)] ')

     % Notas    : [1] Periodo durante el cual casi no hay nadie en casa
     %            [2] El tinaco se llena aprox. --- veces (conteo manual)

     disp(' ')

        eval(['load ' 'y10casa.txt' ' -ascii'])
        datos = y10casa    ;
        code  = ' a.  "Almost-empty house" '  ;
        coda  = ' '         ;

        toffset   =   17.00   ;  % Hora de inicio de la prueba (h)

        diawe     = [ 2:3 9:10 16:17 23:24 ] ; % Fines semana (d)

        diafoot   = [  ] ; % Eventos especiales (d)


        perimetro =   2.700 ;  % Perimetro externo del tinaco (m)
        espesor   =   0.005 ;  % Espesor de la pared del tinaco (m)

        % -------------------------------------------------
        % CALCULA EL AREA DEL TINACO (m2)
          pi    = 3.14159 ;
          diame = perimetro / pi ;
          diami = diame - 2. * espesor ;
          area  = pi * diami^2 / 4.    ;
        % -------------------------------------------------

        tolh      =   0.0050  ; % Cambio significativo de tirante (m)

        tolh      =   0.0025  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


        tolhini   =  -0.00001 ; % Cambio maximo cuando no hay bomba (m)
        tolhfin   =   0.00150 ; % Cambio mínimo cuando arranca la bomba (m)

        klis      =  -1       ; % Codigo para suavizar antes de buscar picos

        % Se define MANUALMENTE las lista de los minima y maxima, porque
        % los datos de nivel son demasiado inestables (cf: problema termico)

          imin_manual = ...
          [ 2534  2574  2626  3200  3954  4587  6752  9578 10737 14051 ...
           14201 14515 15790 17732 19005 20481 21078 22737 22782 24414 ...
           26240 27654 29598 29740 30611 32832 32868 34250 34693 35366 ...
           35500 38361 38672 ] ;

          imax_manual = ...
          [ 2559  2598  2642  3211  3969  4601  6765  9591 10745 14064 ...
```

```
        14212 14530 15804 17747 19024 20495 21100 22761 22803 24427 ...
        26268 27694 29646 29758 30629 32848 32891 34268 34699 35422 ...
        35605 38391 38758 ] ;


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % BUSQUEDA DE EXTREMA AUTOMATICA, SOLO PARA HASTA DIA 19
          klis = 0 ;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% --------------------------------------------------------------------


elseif (nrep == 1)

% --------------------------------------------------------------------
  disp(' Lugar    : Tinaco de casa particular [Serge Tamari] ')
  disp(' Tinaco   : Cilíndrico - capacidad de hasta 450 L    ')
  disp(' Inicio   : Domingo 9 de septiembre 2007 a las 14:31 ')
  disp(' Final    : Miercoles 3 de octubre 2007 a las 8:01   ')
  disp(' Duracion : 584 horas [24.3 dias]                    ')
  disp(' Datos    : 34,162 registros [una lectura / min.]    ')
  disp(' Sensor   : KPsi # 706221 (rango 5 mca) [IMTA 29258] ')

  % Notas    : [1] Se abre la llave de un baño durante dos fines
  %                de semana (dias 7-8 y 21-23) -> ligera fuga
  %            [2] Periodo normal de trabajo
  %            [3] El tinaco se llena aprox. 98 veces (conteo manual)

  disp(' ')

     eval(['load ' 'y07casa.txt' ' -ascii'])
     datos = y07casa    ;
     code  = ' b.  "Occupied house" '  ;
     coda  = ' '         ;

     toffset   =   14.50   ;  % Hora de inicio de la prueba (h)

     diawe     = [ 0 6:7 13:14 20:21 ] ; % Fines semana (d)

     diafoot   = [  ] ; % Eventos especiales (d)


     perimetro =   2.700 ;  % Perimetro externo del tinaco (m)
     espesor   =   0.005 ;  % Espesor de la pared del tinaco (m)

     % ------------------------------------------------
     % CALCULA EL AREA DEL TINACO (m2)
       pi    = 3.14159 ;
       diame = perimetro / pi ;
       diami = diame - 2. * espesor ;
       area  = pi * diami^2 / 4.    ;
     % ------------------------------------------------

     tolh      =   0.0050  ; % Cambio significativo de tirante (m)

     tolh      =   0.0025  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


     tolhini   =  -0.00001 ; % Cambio maximo cuando no hay bomba (m)
     tolhfin   =   0.00050 ; % Cambio mínimo cuando arranca la bomba (m)
     klis      =   0       ; % Codigo para suavizar antes de buscar picos


    % Lista manual de los minima y maxima (para: klis = -1)

       imin_manual = [ ] ;
       imax_manual = [ ] ;
% --------------------------------------------------------------------
```

```
elseif (nrep == 2)

% ----------------------------------------------------------------------
  disp(' Lugar    : Tinaco del edificio 9 del IMTA             ')
  disp(' Tinaco   : Cilíndrico - capacidad de hasta 2,500 L     ')
  disp(' Inicio   : Lunes 17 de marzo 2008 a las 16:45 [aprox.] ')
  disp(' Final    : Viernes 11 de abril 2008 a las 10:32        ')
  disp(' Duracion : 611 horas [25.5 dias]                       ')
  disp(' Datos    : 36,720 registros [una lectura / min.]       ')
  disp(' Sensor   : KPsi # 601842 [rango 3 mca]                 ')

  disp(' ')

  % Notas    : [1] Vacaciones de Semana Santa del 19 de marzo
  %            (a las 15:00) hasta el Domingo 23 de marzo
  %            [2] El tinaco se rellena aprox. 22 veces (conteo manual)

     eval(['load ' 'y08imta.txt' ' -ascii'])
     datos = y08imta    ;
     code  = ' c.  "Office" ' ;
     coda  = ' '         ;

     toffset   =   17.117 ;  % Hora de inicio de la prueba (h)

     diawe     = [ 5:6 12:13 19:20 26:27 ] ; % Fines semana (d)

     diafoot   = [  ] ; % Eventos especiales (d)


     perimetro =   5.000 ;  % Perimetro externo del tinaco (m)
     espesor   =   0.008 ;  % Espesor de la pared del tinaco (m)

     % ------------------------------------------------
     % CALCULA EL AREA DEL TINACO (m2)
       pi    = 3.14159 ;
       diame = perimetro / pi ;
       diami = diame - 2. * espesor ;
       area  = pi * diami^2 / 4.    ;
     % ------------------------------------------------

     tolh      =   0.0050  ; % Cambio significativo de tirante (m)


     tolh      =   0.0034  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


     tolhini   =  -0.00001 ; % Cambio maximo cuando no hay bomba (m)
     tolhfin   =   0.00400 ; % Cambio mínimo cuando arranca la bomba (m)
     klis      =   0       ; % Codigo para suavizar antes de buscar picos


    % Lista manual de los minima y maxima (para: klis = -1)

       imin_manual = [ ] ;
       imax_manual = [ ] ;
% ----------------------------------------------------------------------

elseif (nrep == 3)

% ----------------------------------------------------------------------
  disp(' Lugar    : Edificio privado donde vive R. Alvarez      ')
  disp(' Tinaco   : Rectangular - capacidad mayor a 16,000 L    ')
  disp(' Inicio   : Domingo 24 de Agosto 2008 a las 19:12       ')
  disp(' Final    : Domingo 28 de Septiembre 2008 a las 14:30   ')
  disp(' Duracion : 908 horas [37.8 dias]                       ')
  disp(' Datos    : 54,492 registros [una lectura / min.]       ')
  disp(' Sensor   : KPsi # 601842 [rango 3 mca]                 ')

  disp(' ')

  % Notas    : [1] Regreso a clases el 24 de Agosto 2008
  %            [2] Día nacional el 16 de Septiembre 2008

     eval(['load ' 'y08edif.txt' ' -ascii'])
     datos = y08edif      ;
     code  = ' d.  "Building 0" ' ;
     coda  = ' '           ;

     toffset   =   19.200 ;  % Hora de inicio de la prueba (h)

     diawe     = [ 0 6:7 13:14 20:21 27:28 ] ; % Fines semana (d)

     diafoot   = [  ] ; % Eventos especiales (d)


     largo     =   5.050 ;  % Largo externo del tinaco (m)
     ancho     =   2.700 ;  % Ancho externo del tinaco (m)
     espesor   =   0.190 ;  % Espesor de la pared del tinaco (m)

     % ----------------------------------------------
     % CALCULA EL AREA DEL TINACO (m2)
       area  = (largo-2*espesor) * (ancho-2*espesor) ;
     % ----------------------------------------------

     tolh      =   0.0050  ; % Cambio significativo de tirante (m)

     tolh      =   0.0045  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


     tolhini   =   0.00000 ; % Cambio maximo cuando no hay bomba (m)
     tolhfin   =   0.00010 ; % Cambio mínimo cuando arranca la bomba (m)
     klis      =   11      ; % Codigo para suavizar antes de buscar picos


    % Lista manual de los minima y maxima (para: klis = -1)

       imin_manual = [ ] ;
       imax_manual = [ ] ;
% ----------------------------------------------------------------------


elseif (nrep == 4)

% ----------------------------------------------------------------------
  disp(' Lugar    : Edificio privado "Los Gallos" <A>           ')
  disp(' Tinaco   : Rectangular - capacidad mayor a 12,000 L    ')
  disp(' Inicio   : Miercoles 9 de Junio 2010 a las 15:00       ')
  disp(' Final    : Martes 6 de Julio 2010 a las 09:30          ')
  disp(' Duracion : 642 horas [ 27 dias]                        ')
  disp(' Datos    : 38530  registros [una lectura / min.]       ')
  disp(' Sensor   : KPsi # 601842 [rango 3 mca]                 ')


  disp(' ')

  % Notas    : [1] Copa del mundo de footbal
  %            [2] Vacacciones a finales de junio

     eval(['load ' 'y10a1tin.txt' ' -ascii'])
     datos = y10a1tin      ;
     code  = ' a.  "Building A"  -  WL method ' ;
     coda  = ' '           ;

     toffset   =  15.00 ;  % Hora de inicio de la prueba (h)

     diawe     = [ 3:4 10:11 17:18 24:25 ] ; % Fines semana (d)

     diafoot   = [ 2.37 2.50 8.54 13.37 18.54 ] ; % Eventos especiales (d)


     % ----------------------------------------------
     % CALCULA EL AREA DEL TINACO (m2)
```

```
       area  = 11.630 ;
     % ------------------------------------------------

     tolh      =   0.0050  ; % Cambio significativo de tirante (m)

     tolh      =   0.0040  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


     tolhini   =  -0.00005 ; % Cambio maximo cuando no hay bomba (m)
     tolhfin   =   0.00005 ; % Cambio mínimo cuando arranca la bomba (m)
     klis      =   60      ; % Codigo para suavizar antes de buscar picos


    % Lista manual de los minima y maxima (para: klis = -1)

       imin_manual = [ ] ;
       imax_manual = [ ] ;
% ----------------------------------------------------------------------


elseif (nrep == 5)

% ----------------------------------------------------------------------
  disp(' Lugar    : Edificio privado "Los Gallos" <B>           ')
  disp(' Tinaco   : Rectangular - capacidad mayor a 12,000 L    ')
  disp(' Inicio   : Jueves 10 de Junio 2010 a las 12:00         ')
  disp(' Final    : Martes 6 de Julio 2010 a las 09:30          ')
  disp(' Duracion : 622 horas [26 dias]                         ')
  disp(' Datos    : 37310 registros [una lectura / min.]        ')
  disp(' Sensor   : KPsi # 601841 [rango 10 mca]                ')

  disp(' ')

  % Notas    : [1] Copa del mundo de footbal
  %            [2] Vacacciones a finales de junio

     eval(['load ' 'y10b1tin.txt' ' -ascii'])
     datos = y10b1tin      ;
     code  = 'Building B (2010) - Proposed method' ;
     coda  = ' '           ;

     toffset   =  12.00 ;  % Hora de inicio de la prueba (h)

     diawe     = [ 2:3  9:10 16:17 23:24 ] ; % Fines semana (d)

     diafoot   = [ 1.37 1.50 7.54 12.37 17.54 ] ; % Eventos especiales (d)


    %- ++++++++++++++++++++++++++++++++++++++++++++++
    %- Se desfaza todo de un dia, para coincidir
    %- con la prueba no. 4 (que empezo un dia antes)

        toffset   = toffset + 24 ;
        diawe     = diawe   + 1  ;
        diafoot   = diafoot + 1  ;
    %- ++++++++++++++++++++++++++++++++++++++++++++++



     % ------------------------------------------------
     % CALCULA EL AREA DEL TINACO (m2)
       area  = 11.740 ;
     % ------------------------------------------------

     tolh      =   0.0050  ; % Cambio significativo de tirante (m)

     tolh      =   0.0040  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


     tolhini   =   0.00000 ; % Cambio maximo cuando no hay bomba (m)
     tolhfin   =   0.00001 ; % Cambio mínimo cuando arranca la bomba (m)
```

```
       klis      =   80      ; % Codigo para suavizar antes de buscar picos


      % Lista manual de los minima y maxima (para: klis = -1)

        imin_manual = [ ] ;
        imax_manual = [ ] ;
  % -------------------------------------------------------------------


elseif (nrep == 6)


  % -------------------------------------------------------------------
    disp(' Lugar    : Edificio privado "Los Gallos" <A>           ')
    disp(' Tinaco   : Rectangular - capacidad mayor a 12,000 L    ')
    disp(' Inicio   : Martes 6 de Julio 2010 a las 14:15       ')
    disp(' Final    : Jueves 12 de Agosto 2010 a las 13:00        ')
    disp(' Duracion : 884 horas [37 dias]                         ')
    disp(' Datos    : 53336 registros [una lectura / min.]        ')
    disp(' Sensor   : Keller # 83 [rango 3 mca]                   ')

    disp(' ')

    % Notas    : [1] Vacacicones escolares

       eval(['load ' 'y10a2tin.txt' ' -ascii'])
       datos = y10a2tin      ;
       code  = ' e.  Building I" ' ;
       coda  = ' '           ;

       toffset   =  14.25 ;  % Hora de inicio de la prueba (h)

       diawe     = [ 4:5 11:12 18:19 25:26 32:33 ] ; % Fines semana (d)

       diafoot   = [ 4 5 ] ; % Eventos especiales (d)

       % ------------------------------------------------
       % CALCULA EL AREA DEL TINACO (m2)
         area  = 11.630 ;
       % ------------------------------------------------

       tolh      =   0.0050  ; % Cambio significativo de tirante (m)

       tolh      =   0.0040  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


       tolhini   =  -0.00005 ; % Cambio maximo cuando no hay bomba (m)
       tolhfin   =   0.00005 ; % Cambio mínimo cuando arranca la bomba (m)
       klis      =   60      ; % Codigo para suavizar antes de buscar picos


      % Lista manual de los minima y maxima (para: klis = -1)

        imin_manual = [ ] ;
        imax_manual = [ ] ;
  % -------------------------------------------------------------------

       elseif (nrep == 7)

  % -------------------------------------------------------------------
    disp(' Lugar    : Edificio privado "Los Gallos" <B>           ')
    disp(' Tinaco   : Rectangular - capacidad mayor a 12,000 L    ')
    disp(' Inicio   :       Martes 6 de Julio 2010 a las 14:15         ')
    disp(' Final    : Jueves 12 de Agosto 2010 a las 13:00        ')
    disp(' Duracion : 884 horas [37 dias]                         ')
    disp(' Datos    : 53283 registros [una lectura / min.]        ')
    disp(' Sensor   : Keller # 22 [rango 3 mca]                   ')

    disp(' ')
```

```
     % Notas    : [1] Vacacicones escolares

        eval(['load ' 'y10b2tin.txt' ' -ascii'])
        datos = y10b2tin      ;
        code  = ' f.  Building II" ' ;
        coda  = ' '           ;

        toffset   =  14.25 ;  % Hora de inicio de la prueba (h)

        diawe     = [ 4:5  11:12 18:19 25:26 32:33 ] ; % Fines semana (d)

        diafoot   = [ 4 5 ] ; % Eventos especiales (d)

        % -------------------------------------------------
        % CALCULA EL AREA DEL TINACO (m2)
          area  = 11.740 ;
        % -------------------------------------------------

        tolh      =   0.0050  ; % Cambio significativo de tirante (m)

        tolh      =   0.0040  ; %%% OPTIMIZADO PARA FUGAS (2011-05-07)


        tolhini   =   0.00000 ; % Cambio maximo cuando no hay bomba (m)
        tolhfin   =   0.000001 ; % Cambio mínimo cuando arranca la bomba (m)
        klis      =   80      ; % Codigo para suavizar antes de buscar picos


      % Lista manual de los minima y maxima (para: klis = -1)

          imin_manual = [ ] ;
          imax_manual = [ ] ;
   % --------------------------------------------------------------------


   else

   % --------------------------------------------------------------------
       disp( ' Elegir una opcion valida !... ')
       return
   % --------------------------------------------------------------------

   end



% Parametros internos, para los calculos
% ====================================

%- 'Cero' numérico (calculos en doble precision)

    epsilon = 1e-10      ;



 %- Criterio para definir los tirantes pequeños (= tinaco vacío)
 %- Los tirantes 'pequeños' seran eliminados del archivo de datos crudos

     hsmall = 0.000             ; % Umbral para tirantes pequeños (m)



%- Técnica para extrapolar los consumos durante los suministros

%  1 = Utiliza los flujos anteriores y posteriores al suministro
%      -> Los flujos de consumo sobreestiman el consumo promedio,
%      cuando se extrapolan a un largo periodo...

%  2 = Utiliza el consumo promedio del periodo -> a priori, es la
%      técnica más 'estable', pero habrá que ser atento en los
%      periodos con pocos datos de consumo (cf: cuando se llena
%      un tinaco siempre a las mismas horas)

   kodsum  = 2          ;



%- Criterios para apreciar las estimaciones de fuga    %%% FUGAS %%%


    tolLc   =  1.2     ;  % Criterio de Hansen (1992) %%% p > 0.99 (df = 3)  %%%



    tolR    = -0.90    ;  % Prueba sobre coef correl (cf: explica 80% de la varianza)



%- Opcion para mejorar las graficas (1: 'si')

    kfig   = 1          ;



     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %%% SOLO PARA ANALISIS DE SENSIBILIDAD %%%
     %%% tolh = input(' Valor de <tolh> (m) -> ' );
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


     %%% tolh = 0.0055


% ----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' PARAMETROS DE CALCULO ')
     disp(' ==================== ')

     disp(' ')
     disp([ ' Criterio <Tolh> (m)    = ' num2str(tolh)    ])

     disp(' ')
     disp([ ' Criterio <Klis>        = ' int2str(klis)    ])

     if (klis >= 0)
       disp([ ' Criterio <Tolhini> (m) = ' num2str(tolhini) ])
       disp([ ' Criterio <Tolhfin> (m) = ' num2str(tolhfin) ])
     end

     disp(' ')
     disp([ ' Criterio <hsmall> (m)  = ' num2str(hsmall)  ])

     disp(' ')
     disp([ ' Técnica <Kodsum>      = ' int2str(kodsum)  ])

     disp(' ')
     disp([ ' Criterio <TolLc>       = ' num2str(tolLc)    ]) %%% FUGAS %%%
     disp([ ' Criterio <TolR>        = ' num2str(tolR)     ]) %%% FUGAS %%%
     disp(' ')
     disp([ ' Opción gráfica <Kfig> = ' int2str(kfig)    ])

     disp(' ')


% ----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' PROCESA LOS DATOS CRUDOS ')
```

```
      disp(' ======================== ')



   %  Procesa los datos crudos: (1) Extrae los datos crudos
   % ======================================================


     %- Dimensiones de la tabla de datos

        [n0 ncol] = size(datos)     ; % Renglones (n0) y columnas (ncol)

        if ( ncol ~= 2 )
           disp(' ')
           error(' - FATAL ! No hay dos columnas de datos.')
        end



        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%% SOLO PARA ANALISIS DE SENSIBILIDAD %%%
        %%% dn0 = input(' Reducir numero de datos <dn0> (1-30) -> ' );
        %%% datos = datos(1:dn0:n0,:)   ;
        %%% [n0 ncol] = size(datos)     ;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




     %- Define los vectores de datos crudos

        t0 = datos(:,1)            ; % Tiempo acumulado (dias)
        h0 = datos(:,2)            ; % Tirante en el tinaco (m)


        if ( t0(1) ~= 0 )

           firstday  =  floor(t0(1))             ;
           firsthour =  24 * (t0(1) - firstday)  ;

           disp(' ')
           disp(' - WARNING ! Los tiempos no empiezan desde cero.')
           disp('   Verificar que <toffset> esta bien definido !')

           disp(' ')
           disp(['   Dia  de inicio <archivo> = ' num2str(firstday)  ])
           disp(['   Hora de inicio <archivo> = ' num2str(firsthour) ])
           disp(['   Hora de inicio <toffset> = ' num2str(toffset)   ])

        end


        disp(' ')
        disp([' Numero total de datos crudos = ' int2str(n0) ])



     %- Elimina los datos de tirante pequeños (cf: cuando el tinaco se vacia) @@@

        t0     = t0(h0 >= hsmall)  ; % Tiempo acumulado (dias)
        h0     = h0(h0 >= hsmall)  ; % Tirante en el tinaco (m)

        nsmall = length(h0)        ;

        if ( nsmall ~= n0 )
           disp(' ')
           disp(' - WARNING ! Se eliminan datos de tirante pequeños.')
           disp([' Numero de datos eliminados  = ' int2str(n0-nsmall) ])
           n0 = nsmall ;
           disp(' ')
           disp([' Numero total de datos crudos = ' int2str(n0) ])
        end




   %  Procesa los datos crudos: (2) transforma los datos crudos
   % ==========================================================


     %- Cambio de unidades para los datos crudos

        t0     = t0 - t0(1)               ; % Empieza a 'cero' @@@
        t0     = t0 + (toffset/24.)       ; % Agrega hora de inicio
        t0     = t0 * 24.                 ; % Convierte tiempo en horas @@@

        tdia0  = t0 - 24. * floor(t0/24.) ; % Hora del dia (0-24 h)



     %- Calcula los Volumenes de agua en el tinaco

        v0     = h0 * area                ; % Volumenes (m3)
        v0     = v0 * 1000.               ; % Volumenes (L) @@@


     %- Determina la duracion total de la prueba

        ttotal = max(t0) - min(t0) ; % Duracion de la prueba (h)

        if ( ttotal ~= t0(n0)-t0(1) )
           disp(' ')
           error(' - FATAL ! Error en los datos de tiempo.')
        end


        disp(' ')
        disp([' Duracion total de prueba (h) = ' num2str(ttotal) ' <<<'  ])


     %- Revisa los intervalos de muestreo

        dt0    = t0(2:n0) - t0(1:(n0-1))   ; % Intervalos de tiempo (h)
        dtraw  = mean(dt0)                 ; % Valor promedio (h)


        fdt0   = abs(dt0-dtraw) > (5/3600) ; % Tolerancia de 5 segundos
        ndt0   = sum(fdt0)                 ; % Busca valores estraños


        disp(' ')
        disp([' Intervalo de muestreo    (s) = ' num2str(dtraw*3600) ' <<<' ])


        if (ndt0 > 0)
           disp(' ')
           disp([' - WARNING ! Hay ' int2str(ndt0) ' intervalos raros...' ])
           disp([' - Intervalo mini  (s) = ' num2str(min(dt0)*3600) ])
           disp([' - Intervalo maxi  (s) = ' num2str(max(dt0)*3600) ])
        end



     %- Dias de fin de semana y dias especiales

        nwe     = length(diawe)                ; % Numero de dias
        nfoot   = length(diafoot)              ; % Numero de dias


        % -------------------------------------------------
          clear datos                 ; %%% Ahora memoria !
```

```
        clear dt0 fdto              ; %%% Ahora memoria
      % ------------------------------------------------



% ----------------------------------------------------------------

      disp(' ')
      disp(' ')
      disp(' BUSCA LOS MINIMA Y MAXIMA LOCALES ')
      disp(' ================================= ')


% Busca los extrema locales: (1) inicializa el filtro
% ==================================================


  %- Código para el filtro de los cambios de tirantes
  %  0 : dato descartado
  %  1 : mínimo local
  %  2 : máximo local
  %  3 : dato de consumo guardado
  %  4 : dato de llenado guardado

     f = 0 * ones(n0,1)      ; % Inicializa el filtro



% Busca los extrema locales: (1) busqueda AUTOMATICA
% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


  if (klis >= 0)


        disp(' ')
        disp(' Busqueda automatica de extrema locales...')



% Busca los extrema locales: (1a) busca los 'minimos locales'
% ==========================================================


  %- Suaviza eventualmente los datos, antes de buscar los minimos

     if klis == 0

        hh0 = h0             ; % No suaviza los datos

     else

        hh0 = trilis(h0')    ; % Suaviza una primera vez
        for i = 1 : (klis-1) ; % Suaviza 'klis-1' veces
           hh0 = trilis(hh0) ;
        end
        hh0 = hh0'           ;

     end


  %- Busca 'aumentos abruptos' (cf: cuando se prende la bomba)

     for i = 4 : (n0-2)

        condi1 = hh0(i-3)-hh0(i-2) > tolhini ; % Sin bomba: h decrece
        condi2 = hh0(i-2)-hh0(i-1) > tolhini ; % Sin bomba: h decrece
        condi3 = hh0(i-1)-hh0(i)   > tolhini ; % Sin bomba: h decrece
        condi  = condi1 | condi2 | condi3    ; % Uno de los tres...

        condf1 = hh0(i+1)-hh0(i)   > 0       ; % Con bomba: h aumenta
```

```
         condf2 = hh0(i+2)-hh0(i+1) > tolhfin ; % Con bomba: h aumenta
         condf  = condf1 & condf2             ; % Los dos a la vez...

         conda1 = f(i-3) ~= 1                 ; % Evita un "doble minimo"
         conda2 = f(i-2) ~= 1                 ; % Evita un "doble minimo"
         conda3 = f(i-1) ~= 1                 ; % Evita un "doble minimo"
         conda  = conda1 & conda2 & conda3    ; % Los tres a la vez...

         if (condi & condf & conda)
            f(i) = 1                          ;  % Minimo encontrado
         end

      end


   %- Lista 'provisional' de los minima locales

      f(1)  = 1             ; % Clasifica 'a priori' el primer dato
      f(n0) = 1             ; % Clasifica 'a priori' el ultimo dato

      imin  = find(f==1)    ; % Indices de minima locales
      nmin  = length(imin)  ; % Numero de minima locales


      % ------------------------------------------------
        clear condi1 condi2 condi3  ; %%% Ahora memoria
        clear condf1 condf2         ; %%% Ahora memoria
        clear conda1 conda2 conda3  ; %%% Ahora memoria
        clear condi  condf  conda   ; %%% Ahora memoria
      % ------------------------------------------------



  % Busca los extrema locales: (1b) busca los 'maximos locales'
  % ============================================================


   %- Se define el "maximo local" como siendo el maximo entre dos minimos

      for i = 1 : nmin-1

         n1 = imin(i)   + 1    ; % Inicio de un llenado
         n2 = imin(i+1) - 1    ; % Inicio del siguiente llenado
         xh = h0(n1:n2)        ; % Datos entre dos llenados
         xm = max(xh)          ; % Maximo local entre dos llenados
         k  = find(xh == xm)   ; % Indice de (de los) maxima local(es)
         k  = k(1)             ; % Guarda una sola solucion

         f(n1+k-1) = 2         ; % Maximo encontrado

      end


   %- Lista 'provisional' de los maxima locales

      imax = find(f==2)     ; % Indices de maxima locales
      nmax = length(imax)   ; % Numero de maxima locales




  % Busca los extrema locales: (1c) busca los 'minimos locales'
  % ============================================================

   % NOTA: es una nueva busqueda, porque se cometen
   % pequeños errores cuando se suavizan los datos


   %- Borra la lista anterior de minima locales

      for i = 1 : n0
```

```
        if (f(i) == 1)

           f(i) = 0 ;

        end

     end


  %- Se define el "minimo local" como siendo el minimo entre dos maximos

     for i = 1 : nmax-1

        n1 = imax(i)   + 1    ; % Inicio de un llenado
        n2 = imax(i+1) - 1    ; % Inicio del siguiente llenado
        xh = h0(n1:n2)        ; % Datos entre dos llenados
        xm = min(xh)          ; % Maximo local entre dos llenados
        k  = find(xh == xm)   ; % Indice de (de los) minima local(es)
        k  = k(1)             ; % Guarda una sola solucion

        f(n1+k-1) = 1         ; % Minimo encontrado

     end


  %- Lista 'provisional' de los minima locales

     imin = find(f==1)     ; % Indices de maxima locales
     nmin = length(imin)   ; % Numero de maxima locales



% Busca los extrema locales: (1d) clasifica primer dato y ultimo dato
% ===================================================================


     f(1)  = 1             ; % Clasifica 'a priori' el primer dato
     f(n0) = 1             ; % Clasifica 'a priori' el ultimo dato


  %- Vuelve a clasificar el primer y el ultimo dato del archivo

     if ( h0(1) > h0(imax(1)) )
        f(1)         = 2      ; % Primer dato = cambio a 'maximo'
        f(imax(1))   = 0      ; % Reemplaza el dato
     end

     if ( h0(n0) > h0(imax(nmax)) )
        f(n0)         = 2     ; % Ultimo dato = cambio a 'maximo'
        f(imax(nmax)) = 0     ; % Reemplaza el dato
     end




% Busca los extrema locales: (2) definicion MANUAL de los extremas
% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


 elseif (klis == -1)

        disp(' ')
        disp(' Se utiliza una lista pre-definida de extrema locales...')



% Busca los extrema locales: (2a) datos pre-definidos
% ===================================================

  %- Los vectores que contienen los indices de los minima (imin)
```

```
   %- y maxima (imax) locales deben de estar pre-definidos...

   %- Esto se logra con una grafica de los datos crudos ('plot(i:n0,h0)')
   %- y luego con la función 'axis' (para hacer un 'zoom')
   %- y con la función 'ginput' (para digitalizar)...

         imin = imin_manual  ;
         imax = imax_manual  ;

         nmin = length(imin) ;
         nmax = length(imax) ;


    %- Verifica las listas de minima y maxima

         if (nmin == 0) |(nmax == 0)
            disp(' ')
            error(' FATAL - No hay una lista pre-definida de extrema')
         end

         if (abs(nmin-nmax) > 1)
            disp(' ')
            error(' FATAL - Numeros de minima y maxima incompatibles')
         end


    %- Define los valores del filtro 'f' (1= minimo 2= maximo)

         for k = 1:nmin
            f( imin(k) ) = 1 ;
         end

         for k = 1:nmax
            f( imax(k) ) = 2 ;
         end


 % Busca los extrema locales: (2b) clasifica primer dato y ultimo dato
 % ==================================================================


   %- Verifica que el primer y el ultimo dato aun no son clasificados !

      if (imin(1) <= 1) | (imax(1) <= 1)
         disp(' ')
         errror(' FATAL - No clasificar manualmente el primer dato !')
      end

      if (imin(nmin) >= n0) | (imax(nmax) >= n0)
         disp(' ')
         errror(' FATAL - No clasificar manualmente el ultimo dato !')
      end



   %- Clasifica el primer y el ultimo dato del archivo

      if ( imin(1) < imax(1) )
         f(1)         = 2      ; % Primer dato = 'maximo'
      else
         f(1)         = 1      ; % Primer dato = 'minimo'
      end


      if ( imin(nmin) > imax(nmax) )
         f(n0)        = 2      ; % Ultimo dato = 'maximo'
      else
         f(n0)        = 1      ; % Ultimo dato = 'minimo'
      end
  else

    disp(' ')
    disp(' FATAL - valor de <klis> mal definido ...')

  end




 % Busca los extrema locales: (3) lista definitiva
 % ===============================================


   %- Lista 'definitiva' de los minima y maxima locales

      imin = find(f==1)        ; % Indices de minima locales
      nmin = length(imin)      ; % Numero de minima locales

      imax = find(f==2)        ; % Indices de maxima locales
      nmax = length(imax)      ; % Numero de maxima locales




 % Busca los extrema locales: (4) clasifica el archivo de datos
 % ============================================================


   %- Verifica los numeros de minima y maxima

      if ( abs(nmax-nmin) > 1 )
         disp(' ')
         error( ' - FATAL ! Error en el conteo de extrema.')
      end


      disp(' ')
      disp([' Numero de minimos locales = ' int2str(nmin) ' <<< ' ])
      disp([' Numero de maximos locales = ' int2str(nmax) ])


   %- Clasifica el archivo de datos en 4 categorias (variable "kdata")
   %- y define indices para buscar cambios significativos

      if (f(1) == 1) & (f(n0) == 1)

         kdata = 1 ; % Empieza con minimo, termina con minimo

         kneg  = 0 ;
         dneg  = 1 ;
         kpos  = 1 ;
         dpos  = 0 ;

      elseif (f(1) == 2) & (f(n0) == 2)

         kdata = 2 ; % Empieza con maximo, termina con maximo

         kneg  = 1 ;
         dneg  = 0 ;
         kpos  = 0 ;
         dpos  = 1 ;

      elseif (f(1) == 1) & (f(n0) == 2)

         kdata = 3 ; % Empieza con minimo, termina con maximo

         kneg  = 1 ;
         dneg  = 1 ;
         kpos  = 0 ;
```

```
         dpos  = 0 ;

      elseif (f(1) == 2) & (f(n0) == 1)

         kdata = 4 ; % Empieza con maximo, termina con minimo

         kneg  = 0 ;
         dneg  = 0 ;
         kpos  = 1 ;
         dpos  = 1 ;

      else

         disp(' ')
         error( ' - FATAL ! Error al inicio o al final del archivo.')

      end




% ----------------------------------------------------------------

      disp(' ')
      disp(' ')
      disp(' CONSUMO PROMEDIO ESTIMADO CON LOS EXTREMOS LOCALES ')
      disp(' ================================================== ')



% Opcion: (1) busca decrementos de volumen desde maximo a minimo
% ==============================================================


  %- Busqueda de los indices de los extrema para hacer el calculo

     if (kdata == 1)        ; % Empieza con minimo, termina con minimo

        ipicmax = imax(1 : nmax) ;
        ipicmin = imin(2 : nmin) ;

     elseif (kdata == 2)    ; % Empieza con maximo, termina con maximo

        ipicmax = imax(1 : nmax-1) ;
        ipicmin = imin(1 : nmin) ;

     elseif (kdata == 3)     ; % Empieza con minimo, termina con maximo

        ipicmax = imax(1 : nmax-1) ;
        ipicmin = imin(2 : nmin) ;

     elseif (kdata == 4)     ; % Empieza con maximo, termina con minimo

        ipicmax = imax(1 : nmax) ;
        ipicmin = imin(1 : nmin) ;

     else

        disp(' ')
        error(' FATAL - Error en clasificacion del archivo')

     end


  %- Calculo de "pico a pico"

     sumdv   =  sum( v0(ipicmax) - v0(ipicmin) ) ;
     sumdt   =  sum( t0(ipicmax) - t0(ipicmin) ) ;

     Qmean0  = - sumdv / sumdt ;
  %- Calculo eliminando los picos (porque pueden ser mal estimados)

     sumdv   =  sum( v0(ipicmax+1) - v0(ipicmin-1) ) ;
     sumdt   =  sum( t0(ipicmax+1) - t0(ipicmin-1) ) ;

     Qmean1  = - sumdv / sumdt ;


     disp(' ')
     disp(' Consumo promedio durante la prueba [usando extrema locales] ')
     disp(' ')
     disp(['  - Consumo [pico a pico]  (L/h) = ' num2str(Qmean0) ' <<<' ])
     disp(['  - Consumo [fuera de pico](L/h) = ' num2str(Qmean1) ' <<<' ])


% ----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' EXTRAE LOS DATOS ')
     disp(' ================ ')


% Extrae los datos: (1) busca decrementos significativos ('consumos')
% ===================================================================


  %- Algoritmo de busqueda

     for k = 1 : (nmax-kneg)

        i     = imax(k)           ; % Inicia desde un maximo local
        j     = i + 1             ; % Inicializa 'j'
        jstop = imin(k+dneg) - 1  ; % Termina antes de un minimo


        while (j < jstop) & (j < (n0-1))


            tolhh = tolh          ; % Inicializa criterio estadistico

            while ((h0(i)-h0(j)) < tolhh) & (j < jstop)

               j = j + 1          ; % No hay cambio

             %-------------------------------------------------------
             %- SE VUELVE 'MAS ESTRICTO' PARA LOS FLUJOS PEQUEÑOS
             %- PORQUE LOS DATOS YA NO SON 'BIEN' AUTO-CORRELACIONADOS
             %- CUANDO EL INTERVALO DE TIEMPO ES MAYOR A 2 h

               if ( (t0(j)-t0(i)) > 2.) & ( (t0(j)-t0(i)) <= 6.)

                  tolhh = 1.5 * tolh    ;

               elseif ( (t0(j)-t0(i)) > 6.)

                  tolhh = 2.0 * tolh    ;

               end
             %-------------------------------------------------------

            end

            f(j) = 3              ; % Punto encontrado


           if ( h0(j)-h0(jstop) < tolhh)
               f(j)  = 0          ; % Elimina eventualmente ultimo
           end
```

```
         i    = j               ; % Cambio de indice
         j    = i + 1           ; % Cambio de indice

      end

   end


 %- Lista de los indices encontrados

   ineg = find(f==3)            ; % Indices de consumos (-)
   nneg = length(ineg)          ; % Numero de  consumos (-)



% Extrae los datos: (2) busca incrementos significativos ('suministros')
% ======================================================================


 %- Algoritmo de busqueda

   for k = 1 : (nmin-kpos)

      i     = imin(k)           ; % Inicia desde un minimo local
      j     = i + 1             ; % Inicializa 'j'
      jstop = imax(k+dpos) - 1  ; % Termina antes de un maximo


      while (j < jstop) & (j < (n0-1))

          tolhh = tolh          ; % Inicializa criterio estadistico

          while ((h0(j)-h0(i)) < tolhh) & (j < jstop)

             j = j + 1          ; % No hay cambio

           %-------------------------------------------------------
           %- SE VUELVE 'MAS ESTRICTO' PARA LOS FLUJOS PEQUEÑOS
           %- PORQUE LOS DATOS YA NO SON 'BIEN' AUTO-CORRELACIONADOS
           %- CUANDO EL INTERVALO DE TIEMPO ES MAYOR A 2 h

             if ( (t0(j)-t0(i)) > 2.) & ( (t0(j)-t0(i)) <= 6.)

                tolhh = 1.5 * tolh    ;

             elseif ( (t0(j)-t0(i)) > 6.)

                tolhh = 2.0 * tolh    ;

             end
           %-------------------------------------------------------

          end

          f(j) = 4              ; % Punto encontrado

         if ((h0(jstop)-h0(j)) < tolhh)
             f(j)  = 0          ; % Elimina eventualmente ultimo
         end

         i    = j               ; % Cambio de indice
         j    = i + 1           ; % Cambio de indice

      end

   end


 %- Lista de los indices encontrados

   ipos = find(f==4)           ; % Indices de rellenos (+)
```

```
     npos = length(ipos)         ; % Numero de  rellenos (+)



  % Extrae los datos: (3) filtro de los datos guardados
  % ===================================================


     fg    = f(f>0)             ; % Codigos guardados (1,2,3,4)
     ng    = length(fg)         ; % Numero de valores guardados

     tg    = t0(f>0)            ; % Fechas guardadas (h)
     tdiag = tdia0(f>0)         ; % Horas del dia guardadas
     hg    = h0(f>0)            ; % Tirantes guardados (m)
     vg    = v0(f>0)            ; % Volumenes guardados (L)

     disp(' ')
     disp([' Numero de valores extraidos   = ' int2str(ng)      ])
     disp([' Proporcion de datos extraidos = ' num2str(ng/n0,2) ' <<< ' ])



  % Extrae los datos: (4) duracion de los consumos / suministros
  % ============================================================


     disp(' ')
     disp(' ')
     disp(' ANALIZA LA DURACION DE LOS CONSUMOS Y SUMINISTROS ')
     disp(' ================================================= ')

     tgmin = tg(fg==1)     ; % Fechas de los minimos (h)
     ngmin = length(tgmin) ; % Numero de valores

     tgmax = tg(fg==2)     ; % Fechas de los maximos (h)
     ngmax = length(tgmax) ; % Numero de valores


     if ( (ngmin ~= nmin) | (ngmax ~= nmax) )
        disp(' ')
        error(' - FATAL ! Problema en num. de extremos locales')
     end


     if (kdata == 1)      % Empieza con minimo, termina con minimo

        dtgs  = tgmax(1:ngmax)  - tgmin(1:(ngmin-1))  ;
        dtgc  = tgmin(2:ngmin)  - tgmax(1:ngmax)      ;

     elseif (kdata == 2)  % Empieza con maximo, termina con maximo

        dtgs  = tgmax(2:ngmax)  - tgmin(1:ngmin)      ;
        dtgc  = tgmin(1:ngmin)  - tgmax(1:(ngmax-1))  ;

     elseif (kdata == 3)  % Empieza con minimo, termina con maximo

        dtgs  = tgmax(1:ngmax) - tgmin(1:ngmin)      ;
        dtgc  = tgmin(2:ngmin) - tgmax(1:(ngmax-1))  ;

     elseif (kdata == 4)  % Empieza con maximo, termina con minimo

        dtgs  = tgmax(2:ngmax) - tgmin(1:(ngmin-1))  ;
        dtgc  = tgmin(1:ngmin) - tgmax(1:ngmax)      ;

     end

     tgs     = sum(dtgs)   ;
     tgc     = sum(dtgc)   ;
     tgtotal = tgs + tgc   ;
     if ( (tgtotal-ttotal) > epsilon ) ; % Valores numericamente iguales
        disp(' ')
        error(' - FATAL ! Problema en duracion de la prueba')
     end


    disp(' ')
    disp([' - Duracion de los consumos    (h)  = ' num2str(tgc) ' <<<'    ])
    disp(['   Proporcion temporal         (-)  = ' num2str(tgc/ttotal,2)  ])
    disp(' ')
    disp([' - Duracion de los suministros (h)  = ' num2str(tgs)  ' <<<'    ])
    disp(['   Proporcion temporal         (-)  = ' num2str(tgs/ttotal,2)   ])


% ----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' ANALIZA LOS CAMBIOS SIGNFICATIVOS ')
     disp(' ================================= ')


% Analiza los cambios: (1) calcula los cambios significativos
% ==========================================================


  %- Gradientes en los datos ("datos guardados")
  %- Flujos: positivo para 'consumo', negativo para 'suministro' @@@
  %- NOTA: el simbolo './' representa una division scalar

     nx    =  ng-1                        ; % Numero datos interpolados

     tx    = (tg(2:ng)+tg(1:(ng-1)))/2.     ; % Fechas interpoladas (h)

     tdiax = tx - 24. * floor(tx/24.)       ; % Hora del dia (0-24 h)

     hx    = (hg(2:ng)+hg(1:(ng-1)))/2.     ; % Tirantes interpolados (m)
     vx    = (vg(2:ng)+vg(1:(ng-1)))/2.     ; % Volumenes interpolados (L)

     dtx   = tg(2:ng) - tg(1:(ng-1))        ; % Intervalos de tiempo (h)
     dhx   = hg(2:ng) - hg(1:(ng-1))        ; % Cambios de tirante (m)
     dvx   = vg(2:ng) - vg(1:(ng-1))        ; % Cambio de volumen (L)

     tasax = - dhx ./ dtx                   ; % Tasa cambio tirante (m/h)
     phix  = - dvx ./ dtx                   ; % Flujos calculados (L/h)

     fx    =   (phix >= 0)                  ; % Filtro de los consumos



% Analiza los cambios: (2) estadistica sobre los intervalos de tiempo
% ===================================================================

     disp(' ')
     disp(' Estadistica sobre los intervalos de tiempo (h)   ')
     disp(' ')
     disp(['  - Promedio            = ' num2str( mean(dtx) )       ])
     disp(['  - Desviacion estándar = ' num2str( std(dtx)  )       ])
     disp(['  - Minimo              = ' num2str( statis(dtx,0) )   ])
     disp(['  - Primer cuartil      = ' num2str( statis(dtx,25) )  ])
     disp(['  - Mediana             = ' num2str( statis(dtx,50) )  ])
     disp(['  - Tercer cuartil      = ' num2str( statis(dtx,75) )  ])
     disp(['  - Maximo              = ' num2str( statis(dtx,100) ) ])
     disp(' ')
     disp(['  - Intervalos > 1 hora = ' num2str(sum(dtx>1.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>1.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>1.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 2 hora = ' num2str(sum(dtx>2.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>2.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>2.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 3 hora = ' num2str(sum(dtx>3.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>3.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>3.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 4 hora = ' num2str(sum(dtx>4.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>4.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>4.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 5 hora = ' num2str(sum(dtx>5.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>5.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>5.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 6 hora = ' num2str(sum(dtx>6.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>6.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>6.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 8 hora = ' num2str(sum(dtx>8.))       ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>8.),1)     ])
     disp(['    Proporción          = ' num2str(sum(dtx>8.)/nx,2)  ])
     disp(' ')
     disp(['  - Intervalos > 10 h.  = ' num2str(sum(dtx>10.))      ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>10.),1)    ])
     disp(['    Proporción          = ' num2str(sum(dtx>10.)/nx,2) ])
     disp(' ')
     disp(['  - Intervalos > 12 h.  = ' num2str(sum(dtx>12.))      ])
     disp(['    CANTIDAD            = ' num2str(sum(dtx>12.),1)    ])
     disp(['    Proporción          = ' num2str(sum(dtx>12.)/nx,2) ])




% Analiza los cambios: (3) Cambios de volumen, tirante...
% =========================================================

     wtotalc  = - sum( dvx(fx) )      ; % Total de consumos    (L)
     wdiarioc =   wtotalc/tgc         ; % Consumo promedio horario (L/h)

     wtotals  =   sum( dvx(~fx) )     ; % Total de suministros (L)
     wdiarios =   wtotals/tgs         ; % Suministro promedio horario (L/h)

     wdiff    =   sum(dvx)            ; % Balance absoluto (L)
     wrel     =   wdiff / wtotalc     ; % Balance relativo a consumos (-)


     disp(' ')
     disp(' Consumos durante la prueba [usando cambios significativos] ')
     disp(' ')
     disp(['  - Consumos detectados     (m3) = ' num2str(wtotalc/1000) ])
     disp(['  - Consumo promedio       (L/h) = ' num2str(wdiarioc) ' <<<' ])
     disp(' ')
     disp(['  - Cambio tirante max. (mm/min) = ' num2str( max(tasax(fx))       *1000/60 ) ])
     disp(['  - Cambio tirante <99% (mm/min) = ' num2str( statis(tasax(fx),99) *1000/60 ) ])
     disp(['  - Cambio tirante <95% (mm/min) = ' num2str( statis(tasax(fx),95) *1000/60 ) ])
     disp(['  - Cambio tirante <90% (mm/min) = ' num2str( statis(tasax(fx),90) *1000/60 ) ])
     disp(['  - Cambio tirante medio(mm/min) = ' num2str( mean(tasax(fx))      *1000/60 ) ])
     disp(['  - Cambio tirante min. (mm/min) = ' num2str( min(tasax(fx))       *1000/60 ) ])

     disp(' ')
     disp(' ')
     disp(' Suministros durante la prueba [usando cambios significativos] ')
     disp(' ')
     disp(['  - Suministros detectados  (m3) = ' num2str(wtotals/1000) ])
     disp(['  - Suministro promedio    (L/h) = ' num2str(wdiarios) ' <<<' ])
     disp(' ')
     disp(['  - Cambio tirante max. (mm/min) = ' num2str( max(-tasax(~fx))       *1000/60 ) ])
     disp(['  - Cambio tirante <99% (mm/min) = ' num2str( statis(-tasax(~fx),99) *1000/60 ) ])
     disp(['  - Cambio tirante <95% (mm/min) = ' num2str( statis(-tasax(~fx),95) *1000/60 ) ])
     disp(['  - Cambio tirante <90% (mm/min) = ' num2str( statis(-tasax(~fx),90) *1000/60 ) ])
     disp(['  - Cambio tirante medio(mm/min) = ' num2str( mean(-tasax(~fx))      *1000/60 ) ])
     disp(['  - Cambio tirante min. (mm/min) = ' num2str( min(-tasax(~fx))       *1000/60 ) ])
     disp(' ')
     disp(' ')
     disp(' Balance entre consumos y suministros durante la prueba ')
     disp(' ')
     disp(['  - Diferencia absoluta          (L) = ' num2str(wdiff )     ])
     disp(['    Dif. relativa a los consumos (-) = ' num2str(wrel,2) ' <<<' ])



% ----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' ANALIZA LOS FLUJOS ')
     disp(' ================== ')


% Analiza los cambios: (1) Flujos
% ===============================

  %- Flujos positivos = consumos (L/h)
  %- Flujos negativos = suministros (L/h)

     phic = phix(fx)  ;
     phis = phix(~fx) ;

     disp(' ')
     disp(' Estadistica sobre los flujos positivos = vaciados (L/h)   ')
     disp(' ')
     disp(['  - Promedio            = ' num2str( mean(phic) )       ])
     disp(['  - Desviacion estándar = ' num2str( std(phic)  )       ])
     disp(['  - Minimo              = ' num2str( statis(phic,0) )   ])
     disp(['  - Primer cuartil      = ' num2str( statis(phic,25) )  ])
     disp(['  - Mediana             = ' num2str( statis(phic,50) )  ])
     disp(['  - Tercer cuartil      = ' num2str( statis(phic,75) )  ])
     disp(['  - Maximo              = ' num2str( statis(phic,100) ) ])

     disp(' ')
     disp(' Estadistica sobre los flujos negativos = llenados (L/h)   ')
     disp(' ')
     disp(['  - Promedio            = ' num2str(-mean(phis) )       ])
     disp(['  - Desviacion estándar = ' num2str( std(phis)  )       ])
     disp(['  - Minimo              = ' num2str(-statis(phis,0) )   ])
     disp(['  - Primer cuartil      = ' num2str(-statis(phis,25) )  ])
     disp(['  - Mediana             = ' num2str(-statis(phis,50) )  ])
     disp(['  - Tercer cuartil      = ' num2str(-statis(phis,75) )  ])
     disp(['  - Maximo              = ' num2str(-statis(phis,100) ) ])



% Analiza los cambios: (2) intensidades de (micro)fugas
% ====================================================

     disp(' ')
     disp(' ')
     disp(' ANALIZA LAS FUGAS [= FLUJOS MAS PEQUEÑOS] ')
     disp(' ======================================== ')


  %- Criterios para considerar una posible fuga

     ffuga = fx                             ; % Flujo positivo

     %%% ffuga = fx & ( (tdiax < 6) | (tdiax > 18) )   ;


  %- Lista de los periodos de la prueba
  %- La funcion 'floor' redondea hacia 'abajo'

     disp(' ')
```

```
   disp(' Se determina una fuga por cada periodo de 24 h ...')

   dti    = 24.                          ; % Periodo (24 h)
   txi    = dti * floor(tx/dti)          ; % Fechas redondeadas (h)
   tiini  = min(txi) + dti               ; % Primer periodo
   tifin  = max(txi) - dti               ; % Ultimo perido
   tilist = [tiini : dti : tifin ]       ; % Lista de periodos
   nlist  = length(tilist)               ; % Numero de intervalos


%- Busqueda de las fugas, periodo por periodo



   %- Resetea el grafico (fugas detectadas)  %%% FUGAS %%%

      nfig = 1                     ; %%% FUGAS %%%
      hfig = figure(nfig)          ; %%% FUGAS %%%
      close(hfig)                  ; %%% FUGAS %%%
      figure(nfig)                 ; %%% FUGAS %%%
      hold on                      ; %%% FUGAS %%%

      tit  = code                  ; %%% FUGAS %%%
      titx = 'Hour of the day (h)' ; %%% FUGAS %%%
      tity = 'Stored volume (m3)'  ; %%% FUGAS %%%

      title(tit)                   ; %%% FUGAS %%%
      xlabel(titx)                 ; %%% FUGAS %%%
      ylabel(tity)                 ; %%% FUGAS %%%



   statf  = NaN * ones(nlist,9)          ; % Inicializa  %%% FUGAS %%%
   zmin   = NaN                          ;

   for i = 1 : nlist                     ; % quita un periodo @@@

      fti    = ffuga & (txi==tilist(i))  ; % Filtro por periodo
      z      = phix(fti)                 ; % Flujos guardados
      nz     = length(z)                 ; % Numero de valores
      zmin   = min(z)                    ; % Flujo minimo


      if (zmin == NaN)

         disp(' ')
         disp(' - WARNING - No se encontro fuga durante un periodo...')

      else

         indmin = find((phix==zmin)&fti) ; % Indice del minimo

         if (length(indmin) == 0)        ; % Ningun dato encontrado
            indmin = NaN                 ;
            zmin   = NaN                 ;
            disp(' ')
            disp(' - WARNING - Problema en la busqueda de fugas !...')
         elseif (length(indmin) > 1)     ; % En caso de duplicados
            indmin = indmin(1)           ; % Es arbitrario @@@
            disp(' ')
            disp(' - WARNING - Mas de un minima por dia...')
         end



         % ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         % ANALIZA LA LINEALIDAD DE LA TENDENCIA    %%% FUGAS %%%


         %- Vuelve a los datos crudos
   tbeg  = tx(indmin) - dtx(indmin)/2 ;
   tend  = tx(indmin) + dtx(indmin)/2 ;

   flin    = (t0 >= tbeg) & (t0 <= tend) ;

   tlin    = t0(flin)     ;
   vlin    = v0(flin)     ;
   nlin    = length(tlin) ;

   tdialin = tdia0(flin)  ;
   hlin    = h0(flin)     ;


%- Criterio 1 - Estadistica 'Lc' de Hansen (1992)


    dat = [ vlin tlin ones(nlin,1) ] ;

    [ Lc , Li , R2 , b ] = hansen(dat) ; % b(1)=slope  b(2)=origin

    znew    = -b(1) ;
    errlin0 = 0.5 * abs( (znew-zmin) / (znew+zmin) ) ;


    if ( abs(errlin0) > 0.10 )     ; % Tolerancia 10%
      disp([' - WARNING - REGRESION: Deteccion de fuga dudosa !' num2str(errlin0) ])
    end


 %- Criterio 2 - Estadistica 'Fr' de Fisher (1923)


    coefR    = corrcoef(tlin,vlin) ; % Coeficiente de correlacion lineal
    coefR    = coefR(2) ;

    Fr       = 0.5* (nlin-3).^0.5 * log( (1+coefR)/(1-coefR) ) ;


    errlin0 = 0.5 * abs( (coefR.^2-R2) / (coefR.^2+R2) ) ;

    if ( abs(errlin0) > 0.01 )     ; % Tolerancia 1%
      disp([' - WARNING - HANSEN: Checar calculo de R2 !' num2str(errlin0) ])
    end



  %- Grafico  %%% FUGAS %%%


     if (Lc < tolLc) & (coefR < tolR)


          statf(i,9) = 1         ; % GOOD          %%% FUGAS %%%
          plot(tdialin,vlin/1000,'W-','LineWidth',1.8)

      else

          statf(i,9) = 0         ; % BAD           %%% FUGAS %%%
          plot(tdialin,vlin/1000,'r:','LineWidth',1.8)

      end

 % +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



 statf(i,1)  =  tilist(i)        ; % Inicio del periodo (h)
 statf(i,2)  =  indmin           ; % Indice del minimo
 statf(i,3)  =  zmin             ; % Valor del minimo
 statf(i,6)  =  nz               ; % Numero de valores
```

```
          %%%  statf(i,7)  =  Fr               ; % Fisher (1923)          %%% FUGAS %%%
            statf(i,7)  =  coefR            ; % Coef. Correl. Linear    %%% FUGAS %%%

            statf(i,8)  =  Lc               ; % Hansen(1992)           %%% FUGAS %%%

         end


         if (length(z) >= 2)                ; % Varias 'fugas'

            statf(i,4) = statis(z,1)        ; % Percentil 1%
            statf(i,5) = statis(z,2)        ; % Percentil 2%

         end

      end



 %- Elimina los datos faltantes, para hacer la estadistica de fugas

      ileak = statf(:,2)             ; % Indice temporal de la fuga
      leak  = statf(:,3)             ; % Fuga (flujo minimo de cada dia)

      ileak = ileak(~isnan(leak))    ; % Elimina datos faltantes
      leak  = leak(~isnan(leak))     ; % Elimina datos faltantes


      % ------------------------------------------------
      %  clear ffuga  fti            ; %%% Ahora memoria
      % ------------------------------------------------


      disp(' ')
      disp(' Estadistica global sobre las fugas = FLUJO más pequeño del dia (L/h)   ')
      disp(' ')
      disp(['  - Numero de datos     = ' num2str(length(leak))   ' <<<<<< ' ])

      disp(' ')
      disp(['  - Promedio            = ' num2str(mean(leak)     ) ' <<<<<< ' ])
      disp(['  - Desviacion estándar = ' num2str(std(leak)      ) ' <<<<<< ' ])
      disp(' ')
      disp(['  - Minimo              = ' num2str(min(leak)      ) ' <<<' ])
      disp(['  - Primer cuartil      = ' num2str(statis(leak,25)) ])
      disp(['  - Mediana             = ' num2str(statis(leak,50)) ])
      disp(['  - Tercer cuartil      = ' num2str(statis(leak,75)) ])
      disp(['  - Maximo              = ' num2str(max(leak)      ) ' <<<' ])



   %- SOLO PARA EL ARTICULO (MEJORA FIGURA 1)   %%% FUGAS %%%


   %%%  Caso no. 1
   %%%  esfig(12,20,2,1,0.325,0.355,0.01,0.005,'Hour of the day (d)','Stored volume (m3)',' a.
"Occupied house"  [ U(h) = 5.5 mm ] ',0)

   %%%  Caso no. 4
   %%%   esfig(0,3,1,0.5,5.86,5.96,0.05,0.01,'Hour of the day (d)','Stored volume (m3)',' a.
"Building I"  [ U(h) = 4.0 mm ] ',0)

   %%%  Caso no. 3
   %%%    esfig(0,8,2,1,14.2,15.8,0.4,0.2,'Hour of the day (d)','Stored volume (m3)',' b.
"Building 0"  [ U(h) = 4.5 mm ] ',0)



 %- Elimina los datos que no son confiables   %%% FUGAS %%%

         fleak = statf(:,9) ;
```

```
     LEAK  = statf(fleak,3)         ; % Fugas mas confiables

     disp(' ')
     disp(' Estadistica global sobre las fugas = DATOS MAS CONFIABLES (L/h)   ')

     disp(' ')
     disp(['  - Numero de datos     = ' num2str(length(LEAK))   ' <<< BEST <<< ' ])

     disp(' ')
     disp(['  - Promedio            = ' num2str(mean(LEAK)     ) ' <<< BEST <<< ' ])
     disp(['  - Desviacion estándar = ' num2str(std(LEAK)      ) ' <<< BEST <<< ' ])
     disp(' ')
     disp(['  - Minimo              = ' num2str(min(LEAK)      ) ' <<< BEST ' ])
     disp(['  - Primer cuartil      = ' num2str(statis(LEAK,25)) ])
     disp(['  - Mediana             = ' num2str(statis(LEAK,50)) ])
     disp(['  - Tercer cuartil      = ' num2str(statis(LEAK,75)) ])
     disp(['  - Maximo              = ' num2str(max(LEAK)      ) ' <<< BEST' ])




     disp(' ')
     disp(' Evolucion temporal de los flujos más pequeños (L/h)   ')
     disp(' ')
     disp('       ------------------------------------------------------------------------ ')
     disp('       Hora    Indice   Mini    Perc.   Perc.  Num.    Coef.   Hansen     Ok   ')
     disp('       (h)     del min  (fuga)  1%      2%     valor    R       1992            ')
     disp('       ------------------------------------------------------------------------ ')

     fmt  = '  %i     %i     %5.2f  %5.2f  %5.2f  %i     %5.3f  %5.2f  %i  \n' ;
     disp( sprintf(fmt,statf') )



% -----------------------------------------------------------------

     disp(' ')
     disp(' ')
     disp(' CALCULA LAS FUGAS COMO SI FUERA POR EL METODO MANUAL ')
     disp(' DIFERENCIA DE VOLUMEN ENTRE 1:00 Y 5:00 DE LA MAÑANA ')
     disp(' =================================================== ')


  %- Lista de los periodos de la prueba
  %- La funcion 'floor' redondea hacia 'abajo'

     disp(' ')
     disp(' Se determina una fuga por cada dia...')

     dtm    = 24.                         ; % Periodo (24 h)
     txm    = dtm * floor(t0/dtm)         ; % Fechas redondeadas (h)
     tmini  = min(txm) + dtm              ; % Primer periodo
     tmfin  = max(txm) - dtm              ; % Ultimo perido
     tmlist = [tmini : dtm : tmfin ]      ; % Lista de periodos
     nm     = length(tmlist)              ; % Numero de intervalos


  %- Metodo MTG ('Manual Tank Gauging')

     mleak  = NaN * ones(nm,4)          ; % Inicializa

     tdmi    = 1     ; % Hora de inicio del MTG
     tdmf    = 5     ; % Hora de fin    del MTG


     for i = 1 : nm                     ; % quita un periodo @@@

        fm = (txm == 24*i) & (tdia0 > tdmi) & (tdia0 < tdmf)   ;
        vol    = v0(fm)                 ; % Flujos guardados
        t      = t0(fm)                 ; % Numero de valores
        n      = length(vol)            ;

        if (n < 2)

           fugam = NaN   ;

        else

           fugam  = -(vol(n) - vol(1)) / (t(n) - t(1)) ;

        end

        mleak(i,1)  =  24*i             ; % Dia (expresado en horas)
        mleak(i,2)  =  NaN              ; % Dummy
        mleak(i,3)  =  fugam            ; % Fuga (L/h)

           if (fugam < 0)

              mleak(i,3) = NaN          ; % Elimina (cf: llenado)

           end


        if (n < 2)

           mleak(i,4) = NaN   ;

        else

          mleak(i,4)  =  t(n) - t(1)      ; % Duracion (h)

        end


     end


%- Elimina los datos faltantes, para hacer la estadistica de fugas

%    ileak = statf(:,2)             ; % Indice temporal de la fuga
%    leak  = statf(:,3)             ; % Fuga (flujo minimo de cada dia)

%    ileak = ileak(~isnan(leak))    ; % Elimina datos faltantes
%    leak  = leak(~isnan(leak))     ; % Elimina datos faltantes



     disp(' ')
%    disp(' Estadistica global sobre las fugas = FLUJO más pequeño del dia (L/h)   ')

%    disp(' ')
%    disp(['  - Promedio            = ' num2str(mean(leak)     ) ' <<<<<< ' ])
%    disp(['  - Desviacion estándar = ' num2str(std(leak)      ) ' <<<<<< ' ])
%    disp(' ')
%    disp(['  - Minimo              = ' num2str(min(leak)      ) ' <<<' ])
%    disp(['  - Primer cuartil      = ' num2str(statis(leak,25)) ])
%    disp(['  - Mediana             = ' num2str(statis(leak,50)) ])
%    disp(['  - Tercer cuartil      = ' num2str(statis(leak,75)) ])
%    disp(['  - Maximo              = ' num2str(max(leak)      ) ' <<<' ])

     disp(' ')
     disp(' Evolucion temporal de las fugas estimadas MANUALMENTE (L/h)   ')
     disp(' ')
     disp('        -----------------------------------------------  ')
     disp('        Hora    Indice   Mini      Duracion             ')
     disp('        (h)     (-)      (fuga)    (h)                  ')
     disp('        -----------------------------------------------  ')

     fmt  = '  %i      %i      %5.2f   %5.2f \n' ;
     disp( sprintf(fmt,mleak') )



     disp(' ')
     disp(' ')
     disp(' FIN DE LOS CALCULOS ')
     disp(' ')
     disp(' ')


% ----------------------------------------------------------------



% Grafica (0): Limites de los graficos, caso por caso
% ===================================================


%- NOTA: CUIDADO ! PARA LOS GRAFICOS, SE CAMBIAN LAS UNIDADES !!!

%- General

       xmin =     0.0       ; % Hora del dia (h)
       xmax =    24.0       ;
       dx   =     4.0       ;
       dxx  =     1.0       ;


       dmin =     0.0       ; % Intervalo de tiempo (h)
       dmax =     3.0       ;
       dd   =     1.0       ;
       ddd  =     0.5       ;


%- Casa particular CUERNAVACA (2010)

   if (nrep == 0)

       LD   =     0.5         ; % LIMITE DE DETECCION (L/h)

       tmin =     0         ; % Tiempos (dias)
       tmax =    28         ;
       dt   =     7         ;
       dtt  =     1         ;

       hmin =     0.00      ; % Nivel del agua (m)
       hmax =     0.80      ;

       dh   =     0.20      ;
       dhh  =     0.10      ;

       vmin =     0.00      ; % Volumen del agua (m3)
       vmax =     0.40      ;
       dv   =     0.10      ;
       dvv  =     0.05      ;

       lmin =     0.00      ; % Leaks (L/h)
       lmax =     6.00      ;
       dl   =     2.00      ;
       dll  =     1.00      ;

       zmin =     0.00      ; % Conteo para histograma (-)
       zmax =   100.00      ;
       dz   =    20.00      ;
       dzz  =    10.00      ;


%- Casa particular CUERNAVACA (2007)

   elseif (nrep == 1)
```

```
        LD   =     0.5         ; % LIMITE DE DETECCION (L/h)

        tmin =     0         ; % Tiempos (dias)
        tmax =    28         ;
        dt   =     7         ;
        dtt  =     1         ;

        hmin =     0.240      ; % Nivel del agua (m)
        hmax =     0.80      ;

        dh   =     0.20      ;
        dhh  =     0.10      ;

        vmin =     0.20      ; % Volumen del agua (m3)
        vmax =     0.40      ;
        dv   =     0.10      ;
        dvv  =     0.05      ;

        lmin =     0.00      ; % Leaks (L/h)
        lmax =     6.00      ;
        dl   =     2.00      ;
        dll  =     1.00      ;

        zmin =     0.00      ; % Conteo para histograma (-)
        zmax =   100.00      ;
        dz   =    20.00      ;
        dzz  =    10.00      ;



%- Administracion IMTA (2008)

   elseif (nrep == 2)

        LD   =     1.5         ; % LIMITE DE DETECCION (L/h)

        tmin =     0         ; % Tiempos (dias)
        tmax =    28         ;
        dt   =     7         ;
        dtt  =     1         ;

        hmin =     0.00      ; % Nivel del agua (m)
        hmax =     1.40      ;
        dh   =     0.20      ;
        dhh  =     0.10      ;

        vmin =     0.00      ; % Volumen del agua (m3)
        vmax =     2.50      ;
        dv   =     0.50      ;
        dvv  =     0.10      ;

        lmin =     0.00      ; % Leaks (L/h)
        lmax =     6.00      ;
        dl   =     2.00      ;
        dll  =     1.00      ;

        zmin =     0.00      ; % Conteo para histograma (-)
        zmax =   100.00      ;
        dz   =    20.00      ;
        dzz  =    10.00      ;


%- Edificio MEXICO (2008)

   elseif (nrep == 3)

        LD   =    20         ; % LIMITE DE DETECCION (L/h)

        tmin =     0         ; % Tiempos (dias)
        tmax =    35         ;
```

```
    tmax =    28         ; % PARA COINCIDIR CON LOS OTROS CASOS

       dt   =     7         ;
       dtt  =     1         ;

       hmin =     0.60      ; % Nivel del agua (m)
       hmax =     1.60      ;
       dh   =     0.20      ;
       dhh  =     0.10      ;

       vmin =     8.00      ; % Volumen del agua (m3)
       vmax =    18.00      ;
       dv   =     2.00      ;
       dvv  =     1.00      ;

       lmin =     0.00      ; % Leaks (L/h)
       lmax =   150.00      ;
       dl   =    50.00      ;
       dll  =    10.00      ;

       zmin =     0.00      ; % Conteo para histograma (-)
       zmax =   200.00      ;
       dz   =    50.00      ;
       dzz  =    10.00      ;


%- Edificio "A" JIUTEPEC (Junio 2010)
%- Edificio "B" JIUTEPEC (Junio 2010)

   elseif (nrep == 4) | (nrep == 5)

       LD   =    20         ; % LIMITE DE DETECCION (L/h)

       tmin =     0         ; % Tiempos (dias)
       tmax =    35         ;
       dt   =     7         ;
       dtt  =     1         ;

       hmin =     0.00      ; % Nivel del agua (m)
       hmax =     0.80      ;
       dh   =     0.20      ;
       dhh  =     0.10      ;

       vmin =     0.00      ; % Volumen del agua (m3)
       vmax =    10.00      ;
       dv   =     2.00      ;
       dvv  =     1.00      ;

       lmin =     0.00      ; % Leaks (L/h)
       lmax =   150.00      ;
       dl   =    50.00      ;
       dll  =    10.00      ;

       zmin =     0.00      ; % Conteo para histograma (-)
       zmax =   200.00      ;
       dz   =    50.00      ;
       dzz  =    10.00      ;


%- Edificio "A" JIUTEPEC (Julio 2010)

   elseif (nrep == 6)

       LD   =    20         ; % LIMITE DE DETECCION (L/h)

       tmin =     0         ; % Tiempos (dias)
       tmax =    38         ;
     tmax =    28         ;

       dt   =     7         ;
       dtt  =     1         ;
       hmin =     0.00      ; % Nivel del agua (m)
       hmax =     0.80      ;
       dh   =     0.20      ;
       dhh  =     0.10      ;

       vmin =     0.00      ; % Volumen del agua (m3)
       vmax =    10.00      ;
       dv   =     2.00      ;
       dvv  =     1.00      ;

       qmin = -4000.00      ; % Flujos (L/h)
       qmax =  2000.00      ;
       dq   =  2000.00      ;
       dqq  =  1000.00      ;

       lmin =     0.00      ; % Leaks (L/h)
       lmax =   150.00      ;
       dl   =    50.00      ;
       dll  =    10.00      ;

       zmin =     0.00      ; % Conteo para histograma (-)
       zmax =   200.00      ;
       dz   =    50.00      ;
       dzz  =    10.00      ;


%- Edificio "B" JIUTEPEC (Julio 2010)

   elseif (nrep == 7)

       LD   =    20         ; % LIMITE DE DETECCION (L/h)

       tmin =     0         ; % Tiempos (dias)
       tmax =    38         ;
     tmax =    28         ;

       dt   =     7         ;
       dtt  =     1         ;

       hmin =     0.20      ; % Nivel del agua (m)
       hmax =     0.80      ;
       dh   =     0.20      ;
       dhh  =     0.10      ;

       vmin =     0.00      ; % Volumen del agua (m3)
       vmax =    10.00      ;
       dv   =     2.00      ;
       dvv  =     1.00      ;

       qmin = -4000.00      ; % Flujos (L/h)
       qmax =  2000.00      ;
       dq   =  2000.00      ;
       dqq  =  1000.00      ;

       lmin =     0.00      ; % Leaks (L/h)
       lmax =   150.00      ;
       dl   =    50.00      ;
       dll  =    10.00      ;

       zmin =     0.00      ; % Conteo para histograma (-)
       zmax =   200.00      ;
       dz   =    50.00      ;
       dzz  =    10.00      ;


%- Otros casos

   else

       LD   =    NaN        ; % LIMITE DE DETECCION (L/h)
```

```
        tmin =     0         ; % Tiempos (dias)
        tmax =    35         ;
        dt   =     7         ;
        dtt  =     1         ;

        hmin =     0.60      ; % Nivel del agua (m)
        hmax =     1.80      ;
        dh   =     0.20      ;
        dhh  =     0.10      ;

        vmin =     8.00      ; % Volumen del agua (m3)
        vmax =    18.00      ;
        dv   =     2.00      ;
        dvv  =     1.00      ;

        lmin =     0.00      ; % Leaks (L/h)
        lmax =   150.00      ;
        dl   =    50.00      ;
        dll  =    10.00      ;


    end



% Grafica (2): Tirantes, en función del tiempo
% ==============================================


     %- Resetea el grafico

        nfig = 2             ;
        hfig = figure(nfig)  ;
        close(hfig)          ;
        figure(nfig)         ;
        hold on              ;


     %- Limites del grafico

        tit  = code                      ;
        titx = 'Time (d)'                ;
        tity = 'Water level (m)'         ;


     %- Grafica los datos crudos (tiempo - tirante)

        plot(t0/24,h0,'w-','LineWidth',1.2)

        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Grafica los extremos encontrados

        plot(t0(imin)/24,h0(imin),'r+','LineWidth',1.6,'Markersize',6.0)
        plot(t0(imax)/24,h0(imax),'go','LineWidth',2.2,'Markersize',3.0)


     %- Sobrepone los datos guardados (tiempo - tirante)
     %- NOTA: hacer un 'zoom' para ver detalles...

     %%%   plot(tg/24,hg,'y+','LineWidth',1.2,'Markersize',2.0)
     %%%   plot(tg/24,hg,'y:','LineWidth',1.2)


     %- Mejora la grafica

        if (kfig == 1)
```

```
            esfig(tmin,tmax,dt,dtt,hmin,hmax,dh,dhh,titx,tity,tit,0)
         end



% Grafica (3): Tirantes, en función del tiempo
% ============================================


     %- Resetea el grafico

        nfig = 3             ;
        hfig = figure(nfig)  ;
        close(hfig)          ;
        figure(nfig)         ;
        hold on              ;


     %- Limites del grafico

        tit  = code                     ;
        titx = 'Time (d)'               ;
        tity = 'Water level (m)'      ;


     %- Grafica los datos crudos (tiempo - tirante)

        plot(t0/24,h0,'w-','LineWidth',1.2)

        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Grafica las fugas encontradas
     %- DATOS INTERPOLADOS = un poco distintos de los originales ! @@@

        plot(tx(ileak)/24,hx(ileak),'ro','LineWidth',2.0,'Markersize',5.0)


     %- Sobrepone los datos guardados (tiempo - tirante)
     %- NOTA: hacer un 'zoom' para ver detalles...

     %%%   plot(tg/24,hg,'y+','LineWidth',1.2,'Markersize',2.0)
     %%%   plot(tg/24,hg,'y:','LineWidth',1.2)



     %- Mejora la grafica

        if (kfig == 1)
           esfig(tmin,tmax,dt,dtt,hmin,hmax,dh,dhh,titx,tity,tit,0)
        end


% Grafica (4): Intervalos de tiempo (histograma)
% ==============================================

     %- Resetea el grafico

        nfig = 4             ;
        hfig = figure(nfig)  ;
        close(hfig)          ;
        figure(nfig)         ;
        hold on              ;

     %- Limites del grafico

        tit  = code                     ;
        titx = 'Intervalo de tiempo (h)'  ;
        tity = 'Conteo (-)'               ;
     %- Grafica los datos (histog. intervalos de tiempo)

        hist(dtx,5000)

        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Mejora la grafica

        if (kfig == 1)
           esfig(dmin,dmax,dd,ddd,zmin,zmax,dz,dzz,titx,tity,tit,0)
        end




% Grafica (5): Volumen, en funcion de la hora
%              -> Ubicacion de las 'fugas'
% ==============================================


     %- Resetea el grafico

        nfig = 5              ;
        hfig = figure(nfig)  ;
        close(hfig)           ;
        figure(nfig)          ;
        hold on               ;


     %- Limites del grafico

        tit  = code                       ;
        titx = 'Hour of the day (h)'      ;
        tity = 'Stored volume (m3)'       ;


     % -------------------------------------------------
     %- Elimina unos datos 'para que no se crucen lineas' !

        v00 = v0    ;


        dtu = 2. * dtraw ; % Umbral para eliminar datos

        for i = 1 : n0
           if (tdia0(i) < dtu) | (tdia0(i) > (24-dtu))
              v00(i) = NaN ;
           end
        end

        plot(tdia0,v00/1000,'w-','LineWidth',1.2)
     % -------------------------------------------------



     % -------------------------------------------------
     %- Grafica (datos guardados e interpolados)

        vxx = vx    ;

        for i = 1:nx
           if (tdiax(i) < dtu) | (tdiax(i) > (24-dtu))
              vxx(i) = NaN ;
           end
        end
     %%%   plot(tdiax,vxx/1000,'m-','LineWidth',1.4)
     % ------------------------------------------------


        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Ubica las fugas
     %- DATOS INTERPOLADOS = un poco distintos de los originales ! @@@

        %%% plot(tdiax,vx,'w+')

        plot(tdiax(ileak),vx(ileak)/1000,'ro','LineWidth',3.5,'Markersize',3.0)



     %- Mejora la grafica

        if (kfig == 1)
           esfig(xmin,xmax,dx,dxx,vmin,vmax,dv,dvv,titx,tity,tit,0)
        end



% Grafica (6): Fugas, en funcion del tiempo
% =========================================


     %- Resetea el grafico

        nfig = 6              ;
        hfig = figure(nfig)   ;
        close(hfig)           ;
        figure(nfig)          ;
        hold on               ;


     %- Limites del grafico

        %%% tit  = [code '  -  Flujos = mini, perc. 1% ']   ;

        tit  = code ;
        titx = 'Time (d)'                 ;
        tity = 'Leak (L/h)'    ;



      %- Grafica las fugas (En funcion del tiempo - Todos)


        idia   = statf(:,1) / 24 ;
        fuga   = statf(:,3)      ; % Fuga del dia

        ffuga  = statf(:,9)      ; % Filtro para las fugas 'confiables'


        bar(idia,fuga,'w-')


      %- Marca los datos mas seguros %%% FUGAS %%%

         plot(idia(ffuga),fuga(ffuga),'wo','LineWidth',4.0,'Markersize',5.0)



     %- Grafica el primer percentil de los flujos pequeños

        %%%   plot(idia,statf(:,4),'w--')        % Percentil 1%
        %%% plot(idia,statf(:,5),'g:')           % Percentil 2%


     %- Grafica la fuga promedio

        %%%   y = mean(statf(:,3))               ; % MEDIA
        %%%   plot([tmin tmax],[y y],'w:','LineWidth',2.2)



     %- Grafica el limite de deteccion (LD)

        plot([tmin tmax],[LD LD],'w:','LineWidth',2.2)


     %- Fuga estimada manualmente

     %%%   plot(mleak(:,1)/24,mleak(:,3),'wo','LineWidth',6.0,'Markersize',2.0)
     %%%   plot(mleak(:,1)/24,mleak(:,3),'w--','LineWidth',1.4)



     %- Marca los fines de semana


       %%% plot(diawe,(lmax-dll)*ones(nwe,1),'w+','LineWidth',2.2,'Markersize',8.0)

        plot(diawe,lmax*ones(nwe,1),'w+','LineWidth',4.0,'Markersize',9.0)

        %%% plot(diafoot,(lmax-dll)*ones(nfoot,1),'w+','LineWidth',2.0,'Markersize',7.0)

        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Mejora la grafica

        if (kfig == 1)
           esfig(tmin,tmax,dt,dtt,lmin,lmax,dl,dll,titx,tity,tit,0)
        end



% Grafica (7): Fugas, en funcion del tirante
% ==========================================


     %- Resetea el grafico

        nfig = 7             ;
        hfig = figure(nfig)  ;
        close(hfig)          ;
        figure(nfig)         ;
        hold on              ;


     %- Limites del grafico

        %%% tit  = code    ;

        tit  = code ;
        titx = 'Nivel del agua (m)'                ;
        tity = 'Fuga (L/h)'    ;


      %- Grafica las fugas (En funcion del tiempo - Todos)

         plot(hx(ileak),phix(ileak),'yo','LineWidth',3.5,'Markersize',6.0)

        title(tit)
        xlabel(titx)
        ylabel(tity)


     %- Mejora la grafica

        if (kfig == 1)
           esfig(hmin,hmax,dh,dhh,lmin,lmax,dl,dll,titx,tity,tit,0)
        end



% ------------------------------------------------------------------

   disp(' ')
   disp(' Ok');

% ------------------------------------------------------------------
```