Laplace transform numerical inversion

Bruno Josso - Leif Larsen

1 Laplace transform numerical inversion issue

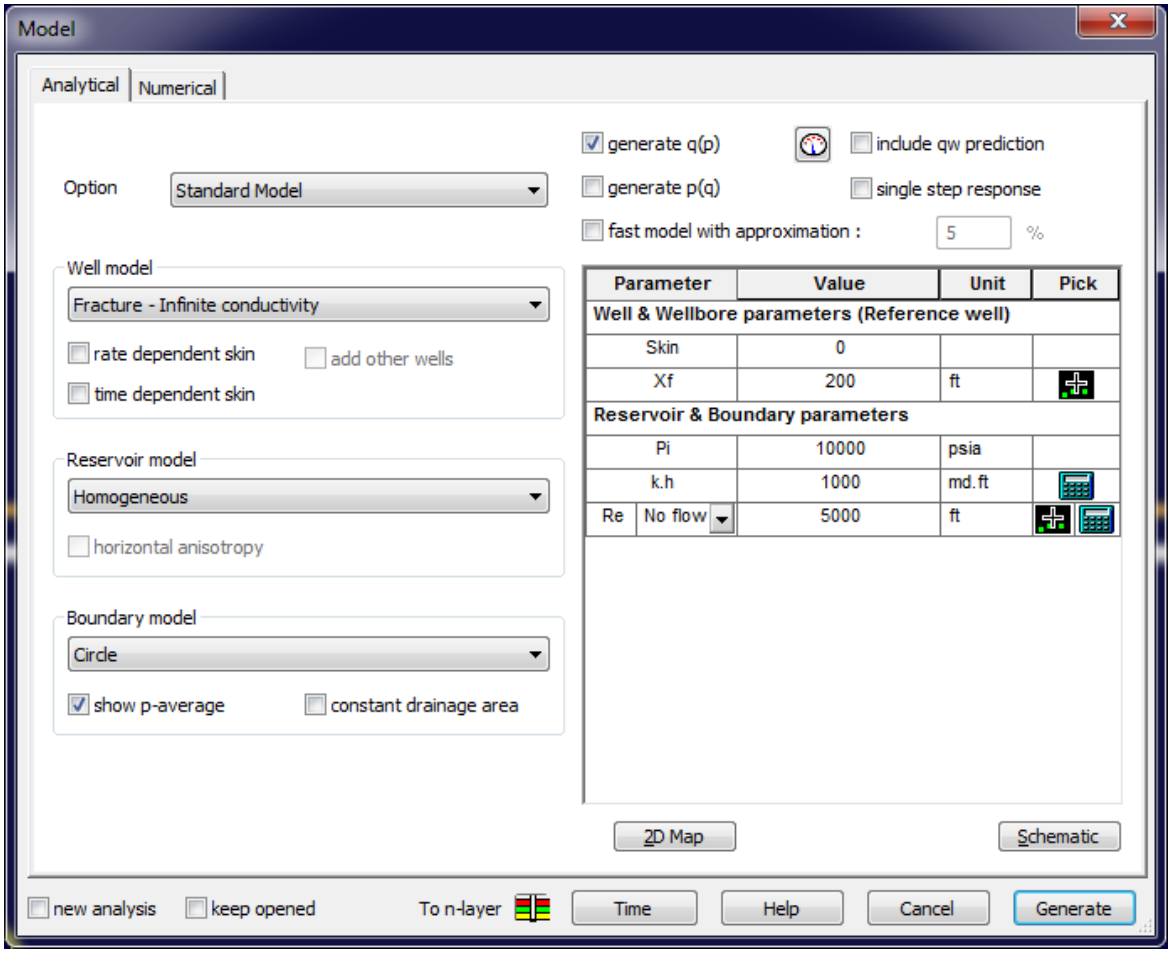
When running an analytical liquid rate simulation on a bounded reservoir an artefact due to Laplace transform numerical inversion algorithm can be noticed. This issue can be illustrated with a simple example: liquid rate simulation on a closed circular homogeneous reservoir with a fracture. The details of the simulation are graphically given in Figure 1(a). The simulation results can be seen in Figure 1(b).

At first, results seem coherent with what can be expected, but if one checks the mean pressure from closer in Figure 1(b) (the yellow curve in the graph at the bottom), one can see that whereas this pressure should monotonously decrease towards a fixed value (around 1000 *psia*) as the reservoir is depleted, it starts to increase slowly between years 2016 and 2017. In Figure 2(a), one can follow the problem from even closer and see that the simulated rate, instead of exponentially decrease towards 0 as expected, becomes negative at the end of 2015. Such results would mean that one swaps from production to injection mode after this date, which is not the case.

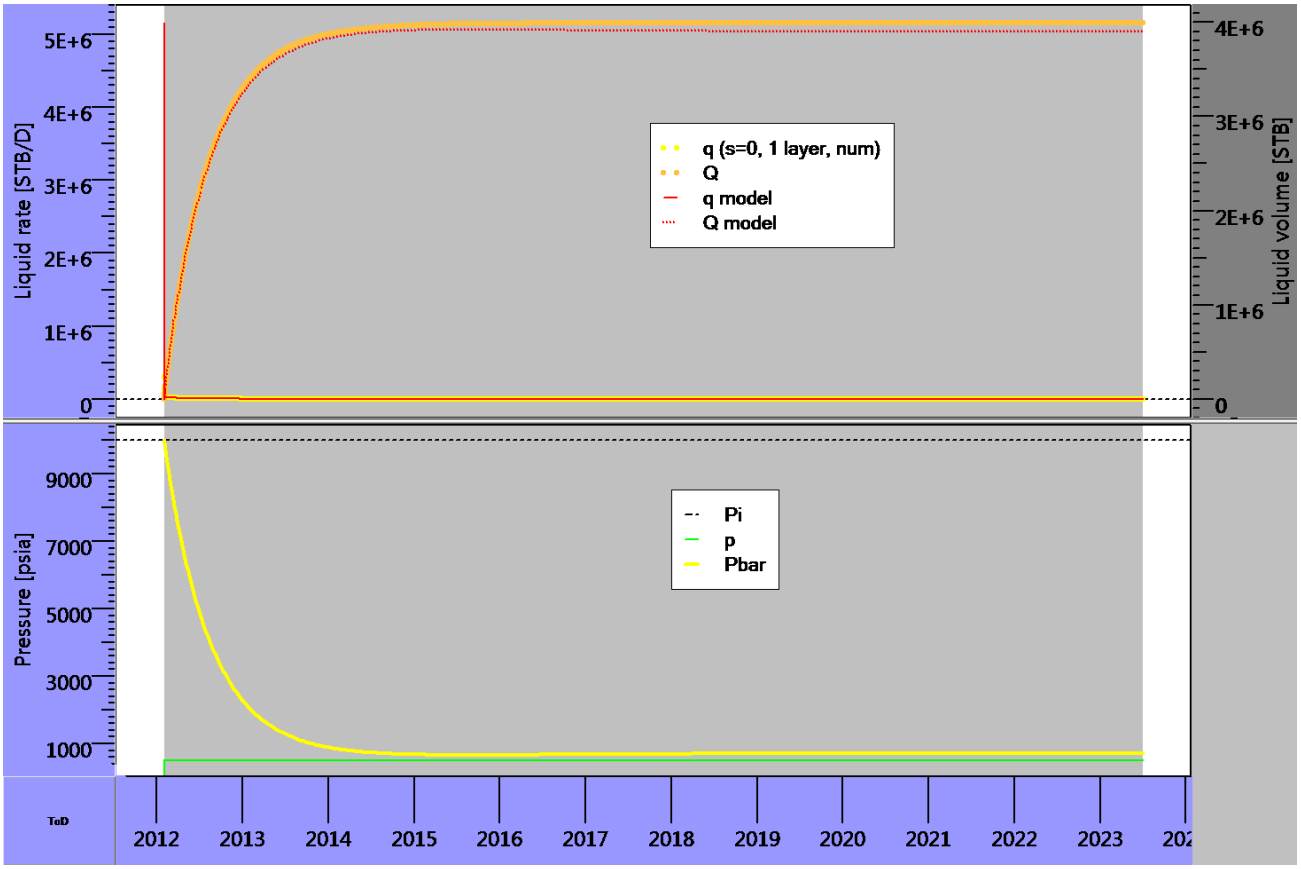
The simulation artefact can also be spotted on the pressure LogLog plot. Hence, it can be seen in Figure 2(b), that at late time, a break appears on the pressure plot (in white). Unfortunately, according to the simulation parameters, one should see a unity slope straight line without any distortion.

In the present case, both simulation and theory do not match. The conclusion is that the simulation gives a wrong result and what is observed here is not a normal behaviour. The simulation is the result of a numerical inversion of a function defined in the Laplace domain. The function is relatively complex here, and we thought at first that the problem was caused either by a bug in the calculation kernel, or by another issue hidden somewhere in Topaze. But it appears that a very similar result can be obtained when inverting an exponential function with a Gaver-Stehfest algorithm taken from an independent implementation.
Indeed: *the problem does directly come from the numerical inversion algorithm*.

After reminding a few definitions regarding the Laplace transform, we will simplify the problem and study several algorithms behaviour when applied on a very simple example. We will eventually see what can be done in the context of reservoir engineering.

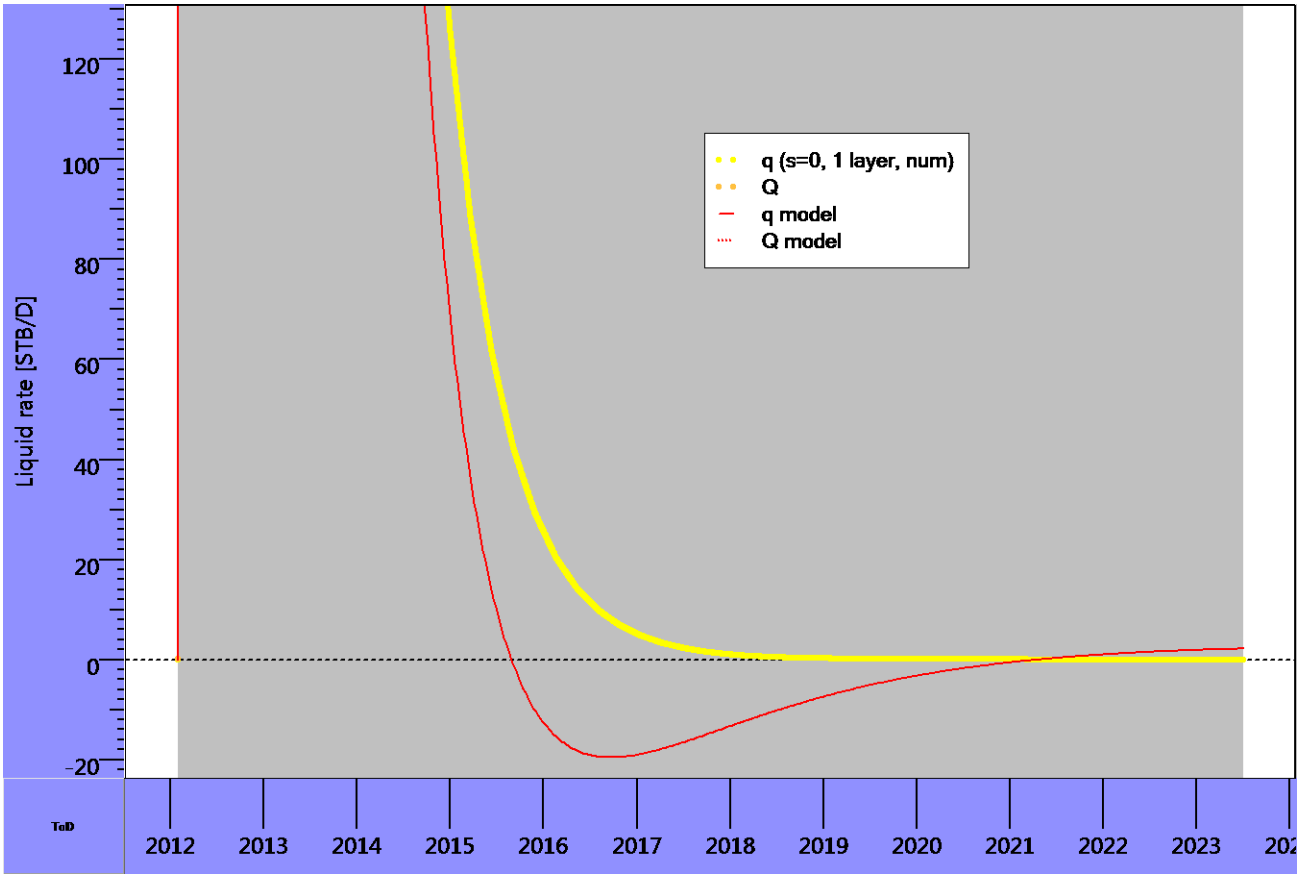


(a) Analytical liquid rate simulation configuration.

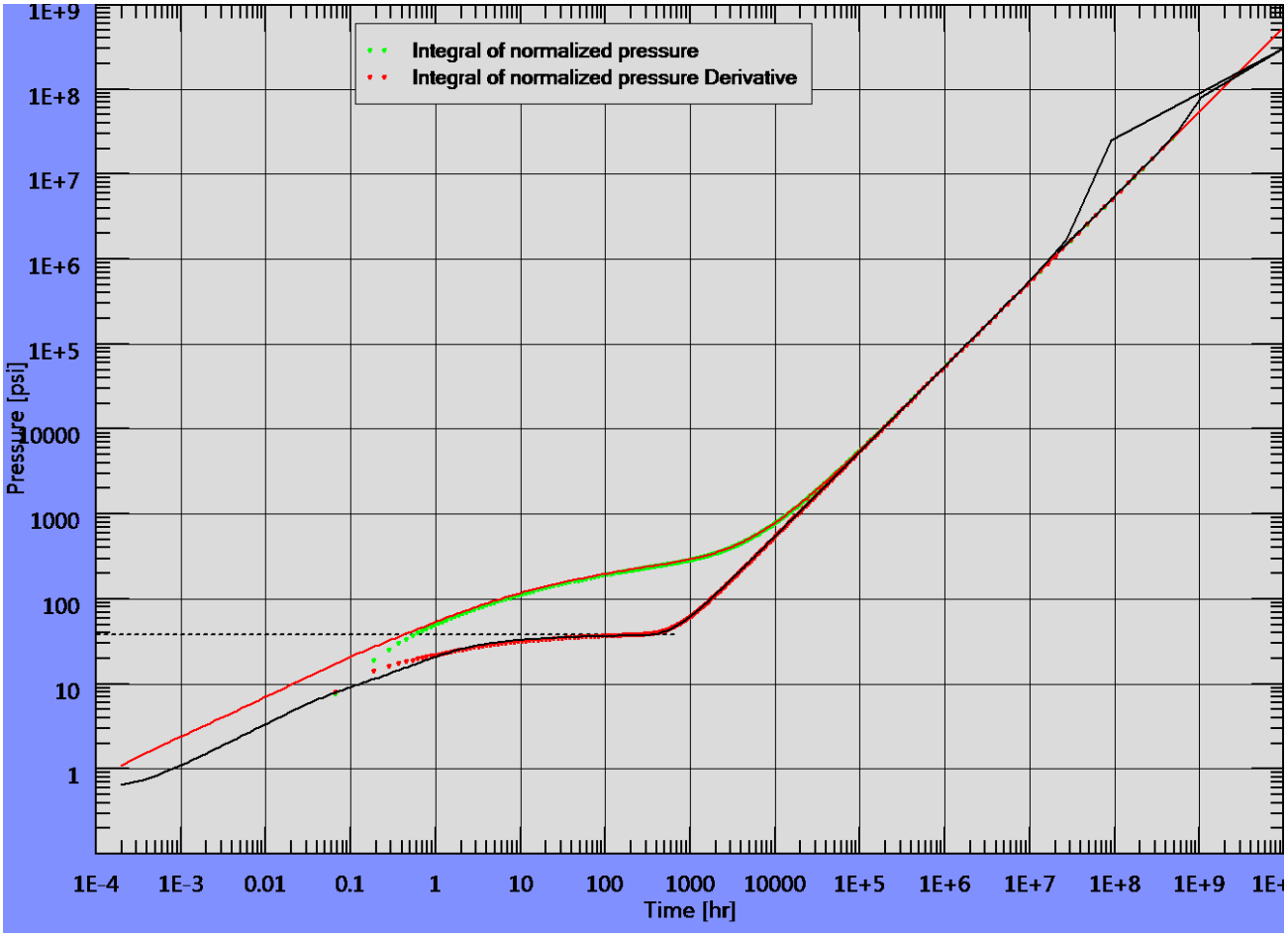


(b) Analytical liquid rate simulation on a bounded reservoir results.

Figure 1: Simulation configuration and results.



(a) Simulation results seen from closer at late time.



(b) Simulation results seen on a LogLog plot

Figure 2: Simulation results.

2 The Laplace transform

2.1 Direct transform

Let $f(t)$ be a function with a real argument $t \in \mathbb{R}$. The bilateral Laplace transform of $f(t)$ is $\mathcal{L}[f(t)] = \hat{F}(p)$, with $p \in \mathbb{C}$ being the Laplace complex argument. The Laplace transform is defined as follows:

$$\hat{F}(p) = \int_{-\infty}^{+\infty} f(t)e^{-pt}dt \tag{1}$$

In practice, one more commonly uses the unilateral Laplace transform expressed as follows[1]:

$$\hat{F}(p) = \int_{0}^{+\infty} f(t)e^{-pt}dt \tag{3}$$

The Laplace argument p can explicitly be written: $p = \sigma + i2\pi\nu$ with $(\sigma, \nu) \in \mathbb{R}^2$. Now, if $\mathcal{FT}[.]$ is the Fourier transform operator, one can then write that the Fourier transform of the function $f(t)e^{-\sigma t}$ is equal to the Laplace transform $\mathcal{L}[f(t)]$:

$$\mathcal{L}[f(t)] = \hat{F}(\sigma + i2\pi\nu) = \int_{-\infty}^{+\infty} \left[f(t)e^{-\sigma t}\right] e^{-2i\pi\nu t}dt = \mathcal{FT}\left[f(t)e^{-\sigma t}\right] \tag{4}$$

Equation (4), is the key point when considering the implementation of the Laplace transform and its inverse. Indeed, very fast algorithms exist that very efficiently perform Discrete Fourier Transform ($\mathcal{DFT}[.]$).

2.2 Inverse transform

The inverse Laplace transform can easily be expressed by referring to the Fourier transform as seen in (4). Hence, if $\mathcal{FT}^{-1}[.]$ is the inverse Fourier transform, one can successively write:

$$\mathcal{FT}\left[f(t)e^{-\sigma t}\right] = \hat{F}(\sigma + 2i\pi\nu) \tag{5}$$

$$f(t) = e^{\sigma t}\mathcal{FT}^{-1}\left[\hat{F}(\sigma + 2i\pi\nu)\right] \tag{6}$$

$$= e^{\sigma t}\int_{-\infty}^{+\infty} \hat{F}(\sigma + 2i\pi\nu)e^{2i\pi\nu t}d\nu \tag{7}$$

One can therefore deduce from expression (7) that the inverse Laplace transform written $\mathcal{L}^{-1}[.]$ can be expressed as follows:

$$\mathcal{L}^{-1}\left[\hat{F}(p)\right] = f(t) = \int_{-\infty}^{+\infty} \hat{F}(\sigma + 2i\pi\nu)e^{(\sigma+2i\pi\nu)t}d\nu \tag{8}$$

$$= \frac{1}{2i\pi}\int_{\sigma-i\infty}^{\sigma+i\infty} \hat{F}(p)e^{pt}dp \tag{9}$$

From expressions (8) and (9), it can be said that the inverse Laplace transform can be calculated by integrating in the complex plane along a vertical line whose abscissa equals $\sigma > 0$. One assumes that there are no singularities on this line.

3 Laplace numerical inversion algorithms testing

3.1 Algorithms

Numerous algorithms can be found in the literature to perform Laplace transform numerical inversion. According to reference [1], a main list of 14 inversion method families can be retained.

[1] If the function $f(t)$ to be processed by unilateral Laplace transform is not equal to 0 for $t < 0$ one therefore calculates the Laplace transform of the product $f(t)u(t)$, with $u(t)$ being the Heaviside step function:

$$\forall t \in \mathbb{R},\ u(t) = \begin{cases} 0 & \text{if} \quad t < 0 \\ 1 & \text{if} \quad t \geq 0. \end{cases} \tag{2}$$

In the present study, stained by our application field, one focuses on a subset of seven algorithms or implementation alternatives.

In the context of reservoir engineering, models are often only known in the Laplace domain for p, the Laplace argument, being a real number: $p \in \mathbb{R}$. Under such a constraint, numerical inversion algorithms that only work with complex Laplace argument ($p \in \mathbb{C}$) cannot be directly considered. Not being able to express reservoir models in the complex plane is the main constraint when selecting a numerical inversion algorithm. Furthermore, one also wants algorithms to be stable, accurate and fast.

The Gaver-Stehfest inversion method, described in [2] and [3], fulfills these criteria in most of the practical cases that are encountered. Indeed, it is highly accurate and stable for most pressure solutions. For this reason, it is one of the most widely used in this domain.

Unfortunately, as could be seen in the first exemple in section (1), Gaver-Stehfest inversion method can also fail and give wrong results under certain circonstances. Indeed, one could see that it suffers limitations for late-time rate data from boundary dominated models produced at constant pressure: late-time exponential decline data cannot be obtained accurately from the Gaver-Stehfest algorithm without some modification.

In this document, we come back to this algorithm, as a reference and a start point, to first see how it can be tuned by using more terms, and second how accuracy can be increased by changing the implementation method.

As mentioned before, when referring to the list of inversion method families given in [1], one can see that about $2/3$ of them, the ones requiring solutions in complex form, are simply not suitable in the current application field. Therefore, in this list only series expansion algorithms could be retained as serious candidates. Three polynomials families have been tested, namely Legendre, Chebyshev and Laguerre. These algorithms respect our main constraint and are based on another principle than the Gaver-Stehfest inversion method.

Since complex numerical inversion methods, i.e. with $p \in \mathbb{C}$, cannot be totally ignored, we have also tested two Fourier transform based algorithms. The first one is directly deduced from expression (4). It belongs to the Dubner-Abate algorithm family [4]. The second one, recently pointed out and very successfully applied in [5], is called Den Iseger's method [6]. Of course, none of these two algorithms can *directly* be applied in the context of reservoir engineering when solutions are not known in complex form. Nevertheless, if one considers that some kind of analytic continuation can be applied on the reservoir models, it is also important to see how well more general algorithms can perform.

A list of seven algorithms has been tested. They are referred to here as :

1. Gaver-Stehfest ($p \in \mathbb{R}$) ➥ see section (4.1.1)
2. Big number Gaver-Stehfest ($p \in \mathbb{R}$) ➥ see section (4.1.2)
3. Legendre polynomials ($p \in \mathbb{R}$) ➥ see section (4.1.3.1)
4. Chebyshev polynomials ($p \in \mathbb{R}$) ➥ see section (4.1.3.2)
5. Laguerre polynomials ($p \in \mathbb{R}$) ➥ see section (4.1.3.3)
6. Fourier ($p \in \mathbb{C}$) ➥ see section (4.2.1)
7. Den Iseger ($p \in \mathbb{C}$) ➥ see section (4.2.2)

3.2 Testing function

Back to the exemple in section (1), both the asymptotic rate $q_D(t_D)$ decline solution and its Laplace transform $\mathcal{L}[q_D(t_D)] = \hat{q}_D(p)$ can be expressed as follows:

$$q_D(t_D) = \Gamma e^{-\mu t} \qquad \xleftrightarrow{\mathcal{L}[.]} \qquad \hat{q}_D(p) = \Gamma \frac{1}{p+\mu} \tag{10}$$

with:

$$\Gamma = \frac{1}{\frac{1}{2}\ln(\frac{4A_D}{e^{\gamma}C_A}) + S} \tag{11}$$

$$A_D = \frac{A}{r_w^2} \tag{12}$$

$$\mu = \frac{2\pi}{A_D \Gamma} \tag{13}$$

A_D denotes a dimensionless area, C_A is the Dietz shape factor and γ denotes Euler's constant.

One will only focus here on this unique simple but illustrative asymptotic rate decline inversion problem. Let $\hat{F}(p) = \frac{\Gamma}{(p+\mu)}$, with $p \in \mathbb{C}$, be a function to be inverted. Ones assumes that $(\mu,\ \Gamma) \in \mathbb{R}_+^{*2}$. As can be found in a Laplace transform dictionary, and as reminded in equation (10), inversion can be performed analytically leading to function $f(t)$:

$$f(t) = \mathcal{L}^{-1}\left[\frac{\Gamma}{(p+\mu)}\right] = \Gamma e^{-\mu t} \qquad \text{with: } t \in \mathbb{R} \tag{14}$$

When applying the "final value theorem" to $\hat{F}(p)$, one can demonstrate that, at late time t, the function $f(t)$ tends towards zero:

$$\lim_{p\to 0} p\hat{F}(p) = \lim_{p\to 0} \frac{\Gamma p}{(p+\mu)} = 0 \tag{15}$$

Furthermore, if $\Gamma > 0$ the function $f(t)$ always remains strictly positive. Therefore, one can write this known results under another form :

$$\lim_{t\to\infty} \Gamma e^{-\mu t} = 0 \tag{16}$$

$$\forall t \in \mathbb{R},\ \Gamma e^{-\mu t} > 0 \tag{17}$$

For a given couple $(\mu,\ \Gamma) \in \mathbb{R}_+^{*2}$, one can see in Figure 3 the function $f(t) = \Gamma e^{-\mu t}$, which is the analytical result of the inversion of $\hat{F}(p) = \frac{\Gamma}{(p+\mu)}$, as well as the result obtained by a usual numerical inversion algorithm. In Figure 3, the analytical solution is called f(t) and it appears in green, whereas the numerical inversion result is called Fp_inv(t) and it appears in blue; both curves nicely overlap.
When zooming on the curves at late time, when they get close to zero, one could see as before on a similar exemple that the reconstructed function resulting from Gaver-Stehfest inversion method becomes negative whereas it should follow an always positive exponential decline.

In the coming sections, one will test the different inversion algorithms on this exponential function $f(t)$ given in (14) and perform such a close late time inspection. One will examine how some Laplace numerical inversion methods converge towards the solution to see whether this inversion artefact can be avoided.

Beyond the numerical inversion accuracy, one will particularly focus on the fact that for all $t \in \mathbb{R}$, the exponential function $f(t)$ should remain strictly positive. The target here is to exhibit a numerical inversion method that accurately inverts $\hat{F}(p) = \frac{\Gamma}{(p+\mu)}$ and always gives strictly positive results.

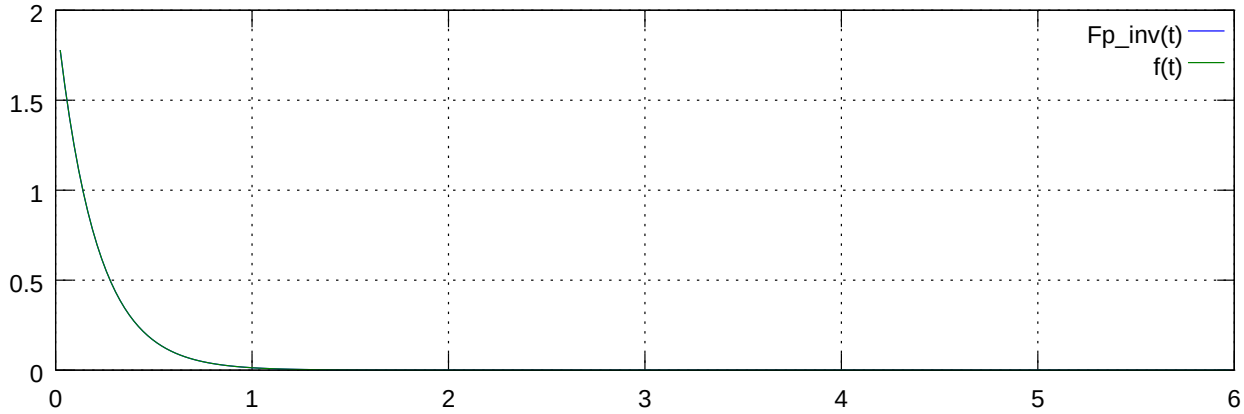


Figure 3: Analytical solution $f(t) = \Gamma e^{-\mu t}$ and numerical inversion result.

4 Inversion algorithms testing

In this section one presents the results obtained by each tested algorithm introduced in section (3). Algorithm comparative tests were only performed on a single function $f(t)$ given in (14) and represented in Figure 3. For clarity sake, results are only presented here in terms of reconstruction errors and shown in a single figure for each algorithm configuration.

In each figure, the reconstruction error, which is equal to the known analytical result minus the inverted Laplace function, appears in blue and is therefore tagged: Error: f(t) - Fp_inv(t). Since, with the chosen test function $f(t)$, one particularly focuses on the sign of the Laplace inverted fonction, one can also see in green, superimposed on the error, the result of a "sign test" tagged: Test Fp_inv(t) < 0. This often crenellated function allows one to easily localise instants for which exponential reconstructed function becomes negative. The test function is positive when the inverted function is positive and negative otherwise. This test function is normalised in magnitude to always keep the same proportion than the reconstruction error.

Algorithms can often be tweaked to increase inversion accuracy. This was not the aim of the study. Therefore, the error magnitude that appears on the graphes should mainly be considered as an indication and not as a method accuracy measurement.

Ideally, with the retained test function, a perfect inversion algorithm would yield a very accurate inverted function with an always positive sign. In this case, the sign test function would be a continuous horizontal line above the time axis. At least, a minor function preconditioning, should allow one to reject at late time the instant when the sign indicator becomes negative. These are the retained criteria to assess the different algorithms.

4.1 Algorithms with real Laplace argument: $p \in \mathbb{R}$

4.1.1 Gaver-Stehfest algorithm

One of the most popular inversion algorithm, especially in reservoir engineering, where one needs to have the Laplace argument p to be real ($p \in \mathbb{R}$), is the Gaver-Stehfest method described in [2] and [3]. This algorithm is fast and usually gives good results, especially for smooth functions. Gaver-Stehfest algorithm is based on the following approximation:

$$\forall t \in \mathbb{R}_+^*, \quad f(t) \approx \frac{\ln(2)}{t} \sum_{n=1}^{N} K_n \hat{F}\left(\frac{n \ln(2)}{t}\right) \tag{18}$$

Parameter N, referred to as the "Stehfest number" should be even. The weighting coefficients K_n can be calculated once and for all during a pre-processing, because they only depend on the Stehfest number N.

As could be seen in the introduction, though this algorithm usually performs well, it fails at late time when inverting the test function $\hat{F}(p) = \frac{\Gamma}{(p+\mu)}$. Indeed, as can be seen in Figure 4, the inverted function oscillates around the analytical solution $f(t)$ and it unfortunately becomes negative from around $t = 3.3$. Indeed, at this instant the sign test indicator becomes negative. One reminds that an appropriate error behaviour would have been to remain close to zero with a sign test indicator always being strictly positive.

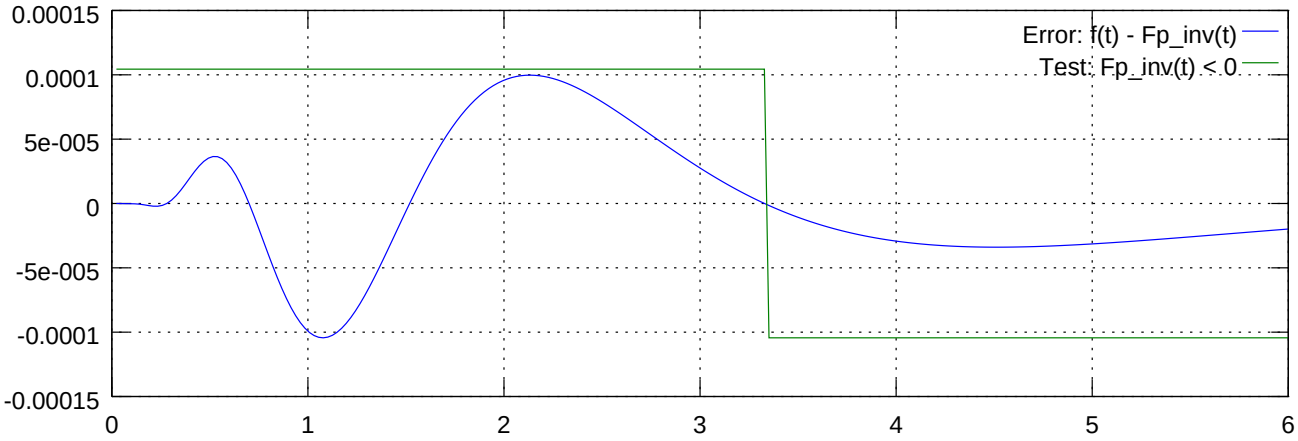


Figure 4: Inversion of $f(t) = e^{-\mu t}$ by Stehfest $N = 14$ algorithm.

The precision of the Gaver-Stehfest inversion method depends on the Stehfest number N. Indeed, one can see in equation (18) that the inversion is based on a summation of N weighted values. Theoretically, the bigger N, the more precise the inversion (but the slower). Results shown in Figure 4 were obtained by choosing $N = 14$. In practice since speed is also important the default Stehfest number is often chosen around $N = 8$. The artefact in therefore more important.

At this point, one can assume that, to get closer to the analytical solution, one only needs to use a bigger Stehfest number N. This actually is what is recommended in the literature. Therefore, several inversion attempts are tested with bigger Stehfest number N:

- One can see in Figure 5 the result obtained with $N = 16$.
  ➥ The reconstruction error is smooth and, as expected, smaller than with $N = 14$.
- In Figure 6, one sets $N = 18$.
  ➥ The reconstruction error is smaller than with $N = 16$, but it becomes "noisy". This means that the reconstruction error is now due to two different phenomena:
  1. algorithm reconstruction error,
  2. numerical approximation error.
- In Figure 7, one sets $N = 20$.
  ➥ The numerical approximation noise becomes predominant, reconstruction error is now bigger than with $N = 18$!
- Finally, in Figure 8, one sets $N = 22$.
  ➥ One can see that the numerical approximation error grows dramatically and highly impacts the reconstruction result.

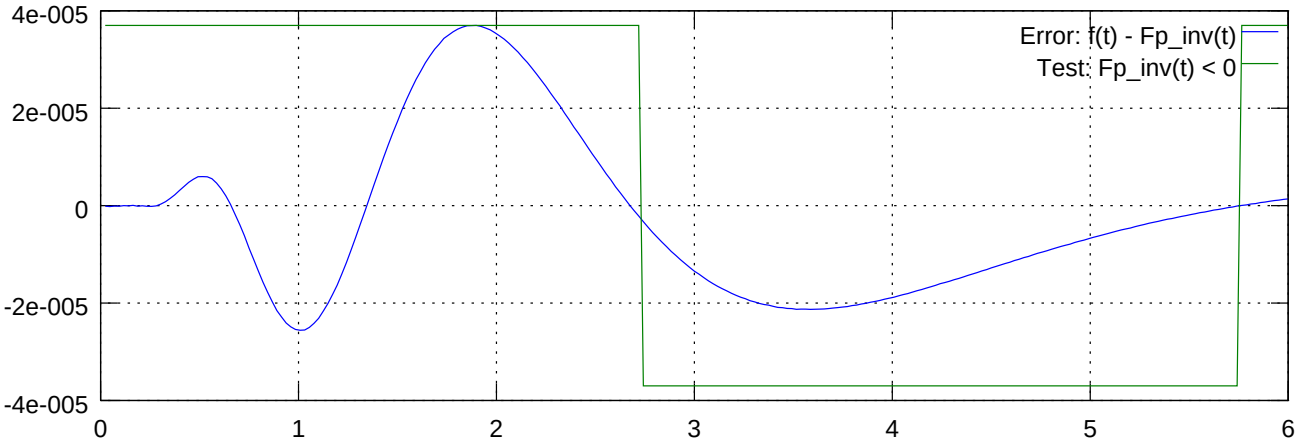


Figure 5: Inversion of $f(t) = e^{-\mu t}$ by Stehfest $N = 16$ algorithm.

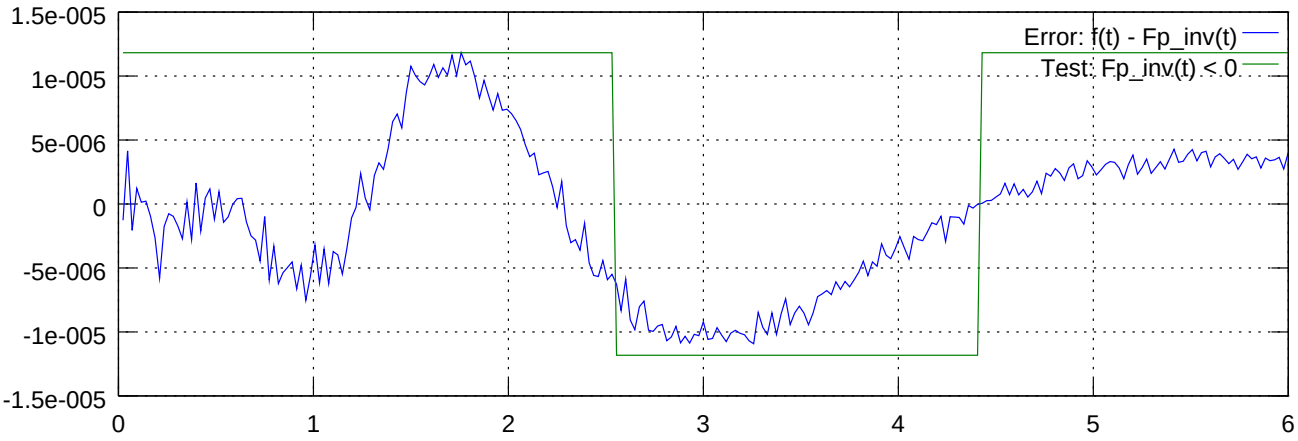


Figure 6: Inversion of $f(t) = e^{-\mu t}$ by Stehfest $N = 18$ algorithm.

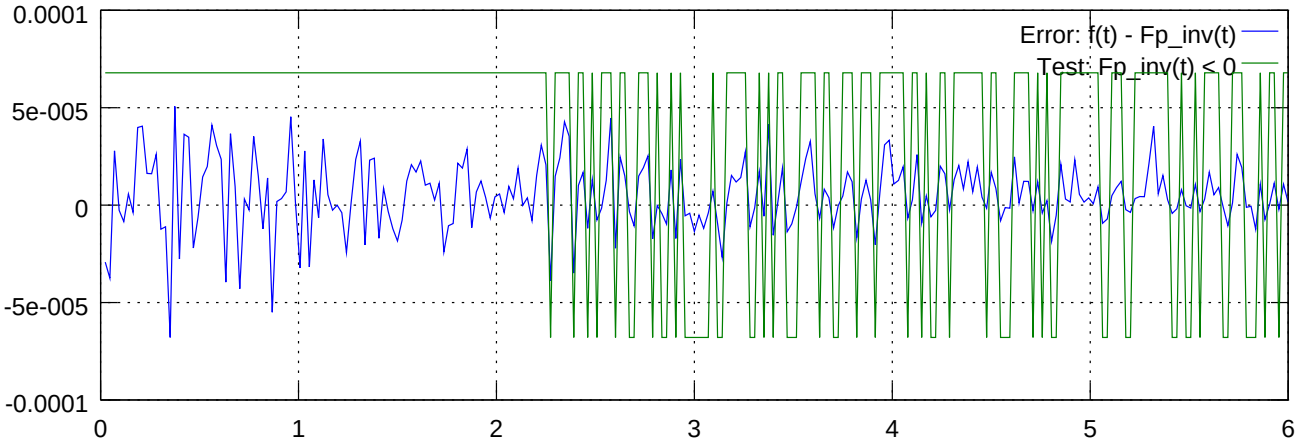


Figure 7: Inversion of $f(t) = e^{-\mu t}$ by Stehfest $N = 20$ algorithm.

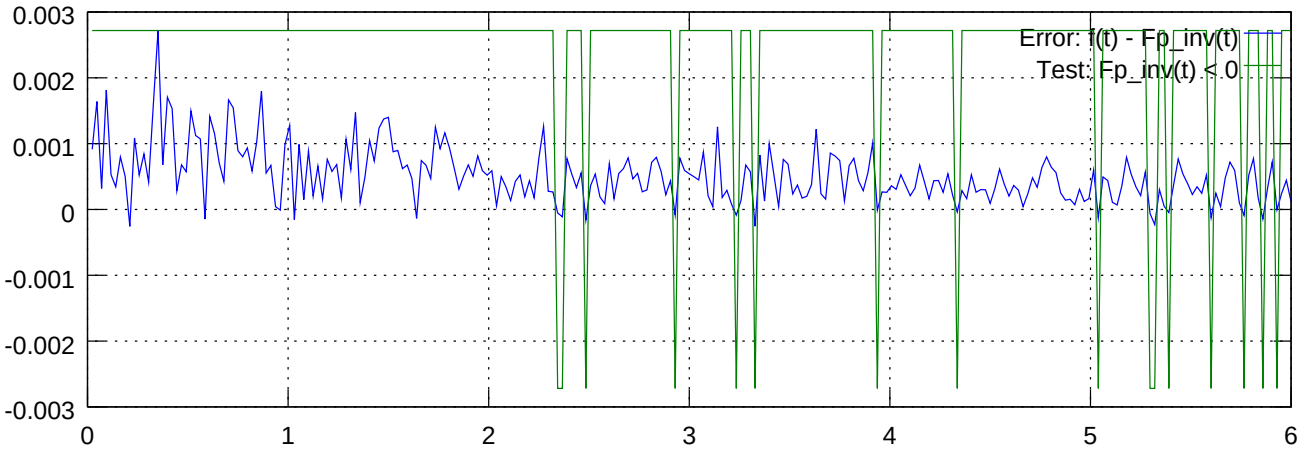


Figure 8: Inversion of $f(t) = e^{-\mu t}$ by Stehfest $N = 22$ algorithm.

In a standard implementation, the Stehfest number N should be adjustable. One can notice here that it is reasonable to only allow a Stehfest number $N \leq 18$. Indeed, if one tries to run Stehfest algorithm with N bigger than 18 by using standard type formats (up to "*long double*" real data type), one will suffer numerical approximation errors. Hence, when the Stehfest number N becomes too big, the result is opposite to what was expected: reconstruction becomes less accurate. To overcome this numerical approximation problem, one has to change the implementation method by using: "arbitrary precision floating point arithmetic". With such an implementation Gaver-Stehfest algorithm can then be pushed further. It can then be called: "big number Gaver-Stehfest algorithm".

4.1.2 Big number Gaver-Stehfest algorithm

What we call the "big number Gaver-Stehfest algorithm" corresponds to the Gaver-Stehfest inversion algorithm for which the Stehfest number N typically becomes greater than 18. One could see in section (4.1.1) that though in standard implementation "*long double*" type is coded with 80 bits, numerical approximation errors occur starting from $N = 18$. Now, if one wants to go further with running the algorithm with $N > 18$, one needs to use *arbitrary precision floating point arithmetic*.

Even when changing the floating point arithmetic, Gaver-Stehfest algorithm is implemented exactly the same way as before. One keeps using the approximation (18) to inverse the Laplace transform. It should also be pointed out that in the reservoir engineering context, one also needs to reimplement the reservoir models by also using the arbitrary precision floating point arithmetic. Only changing the algorithm but keeping the reservoir models old implementation does not solve the numerical approximation problem.

One can see in Figure 9, the big number Gaver-Stehfest algorithm reconstruction errors and sign test for $N = 22,\ 26,\ 30$ and 40 (to be compared with graphes from Figure 4 to Figure 8).
One can see in Figure 9 that with such an implementation, reconstruction behaviour becomes as expected: the bigger N the more accurate the reconstruction and the smaller the error. It can virtually be made as small as desired. It can also be noticed that reconstruction error is smooth; direct comparison can be made for $N = 22$. This means that, with such an arbitrary precision floating point arithmetic implementation, results are not impacted by numerical precision any more.

When checking sign test curves in green in Figure 9, one can see that even if reconstruction error decreases with N, one can never avoid the inverted function to become negative. Depending on N, the negative swing occurs at different instants. Unfortunately, this is not straightforwardly controllable and therefore cannot easily be rejected towards very late time. Generalising this result, one can say that for all retained Stehfest number N, the reconstructed function always

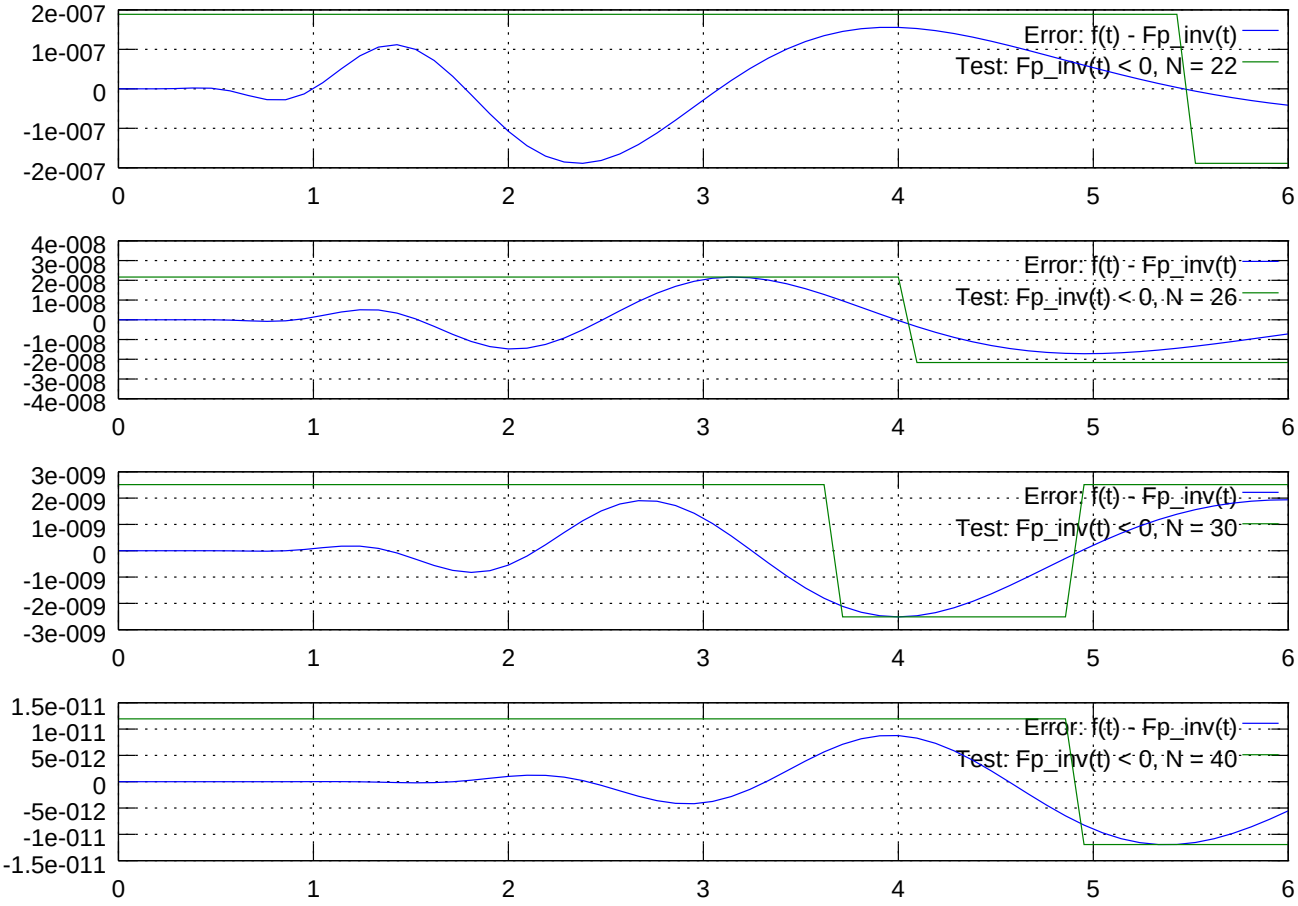


Figure 9: Inversion error of $f(t) = e^{-\mu t}$ by Stehfest $N = 22,\ 26,\ 30$, and 40 algorithm.

oscillates around the true solution that dies towards zero and, in our example, occasionally becomes negative.

As a conclusion, big Stehfest number algorithm can contribute to minimise reconstruction error, but it cannot be retained as an alternative for the current issue. Furthermore, as mentioned before, big Stehfest number algorithm requires the use of an arbitrary precision floating point arithmetic implementation. This is not costless since the inversion algorithm as well as all the Laplace reservoir models need to be rewritten by using this high precision arithmetic to avoid numerical approximation errors within the whole inversion process.

4.1.3 Polynomials series expansion algorithms

The principle of polynomials series expansion algorithms is to approximate the inverted function by a weighing sum of particular polynomials. One has to calculate a given number $K \in \mathbb{N}$ of weighting coefficients α_k that depend on the function to be inverted. In the Laplace domain, let $\hat{F}(p)$ be the function to be inverted and $f(t)$ be the result of this operation. The principle of the series expansion method is to express $f(t)$ as follows:

$$f(t) = w(t)\left(\sum_{k=0}^{+\infty} \alpha_k \Phi_k(t)\right) \approx w(t)\left(\sum_{k=0}^{K-1} \alpha_k \Phi_k(t)\right) \tag{19}$$

$w(t)$ is a windowing function. Depending on the case, $\Phi_k(t)$ can be either a function of Legendre polynomials, Chebyshev polynomials of second kind or Laguerre polynomials. These three alternatives are illustrated in the following sections. Weighting coefficients α_k depend on the values taken by $\hat{F}(p)$ for a limited number of p values. The Laplace argument p is here considered as being a real number.

In practice, the series expansion given in (19) is performed on a limited number of terms K. In theory, the bigger the integer K, the more accurate the inversion process yielding to $f(t)$. In the current implementation the values retained corresponded to the Stehfest number used in section (4.1.1). On the other hand, when the number of coefficients K becomes too big, numerical approximation problems occur. Again, an extra implementation effort should be made in order to get a very accurate reconstruction linked to a big number of weighing coefficients. The situation is strictly similar here as it was with Gaver-Stehfest algorithm.

4.1.3.1 Legendre polynomials

A description of this algorithm can be found in [1]. For Legendre polynomials, weighing coefficients α_k decrease slowly. Therefore, one has to choose K big enough to maximise inversion accuracy. Comparatively, one recalls that for Gaver-Stehfest algorithm, a standard Stehfest number value is typically $N = 8$.

In Figure 10, one can see the reconstruction error obtained when inverting $\hat{F}(p)$, the Laplace transform of function $f(t) = e^{-\mu t}$ by using Legendre algorithm. It oscillates, especially at early time, but error magnitude remains fairly stable. Unfortunately, the way the reconstructed function converges towards the exact solution still causes problem. Indeed, one can see that at some point, the exponential function becomes negative.

4.1.3.2 Chebyshev polynomials

A description of this algorithm can be found in [7]. In Figure 11, one can see the reconstruction error obtained when inverting $\hat{F}(p)$, the Laplace transform of function $f(t) = e^{-\mu t}$ by using Chebyshev algorithm. It oscillates and increases with time. Inversion accuracy is one of the worst among the tested algorithms. One can also see that the reconstructed function eventually becomes negative.

4.1.3.3 Laguerre polynomials

A description of this algorithm can be found in [7]. In Figure 12, one can see the reconstruction error obtained when inverting $\hat{F}(p)$, the Laplace transform of function $f(t) = e^{-\mu t}$ by using Laguerre algorithm. It oscillates but decreases with time. Unfortunately the inverted function occasionally becomes negative.

For further investigation, it should be mentioned that Laguerre polynomials algorithm can be extended by considering a complex Laplace argument $p \in \mathbb{C}$, by opposition to the real complex argument $p \in \mathbb{R}$ scheme tested here. This latter strategy is the one retained by Weeks [8].

4.1.3.4 Conclusion on polynomials series expansion algorithms

As could be seen from section (4.1.3.1) to (4.1.3.3) none of polynomials series expansion algorithms could give a satisfying inversion result. They do not perform better than the standard Gaver-Stehfest reference method. Furthermore, inverted test function always become negative at some points whereas it should always remain strictly positive.

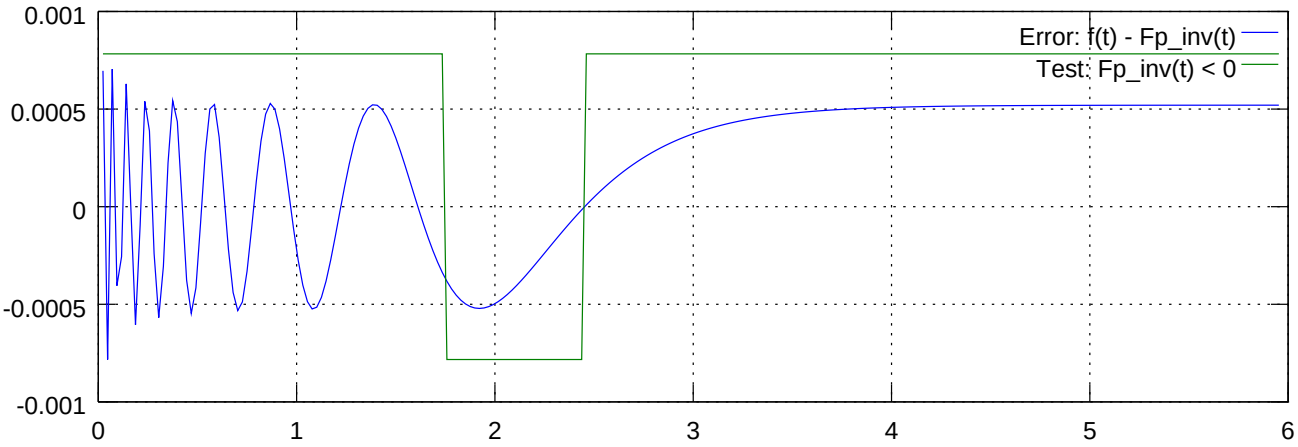


Figure 10: Inversion of $f(t) = e^{-\mu t}$ by Legendre ($K = 20$) algorithm.

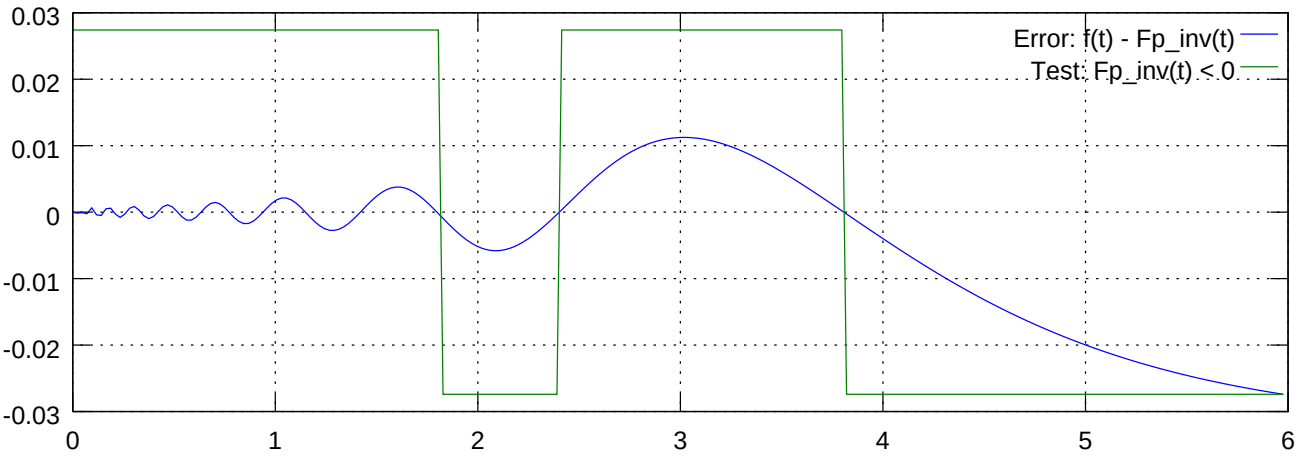


Figure 11: Inversion of $f(t) = e^{-\mu t}$ by Chebyshev ($K = 20$) algorithm.

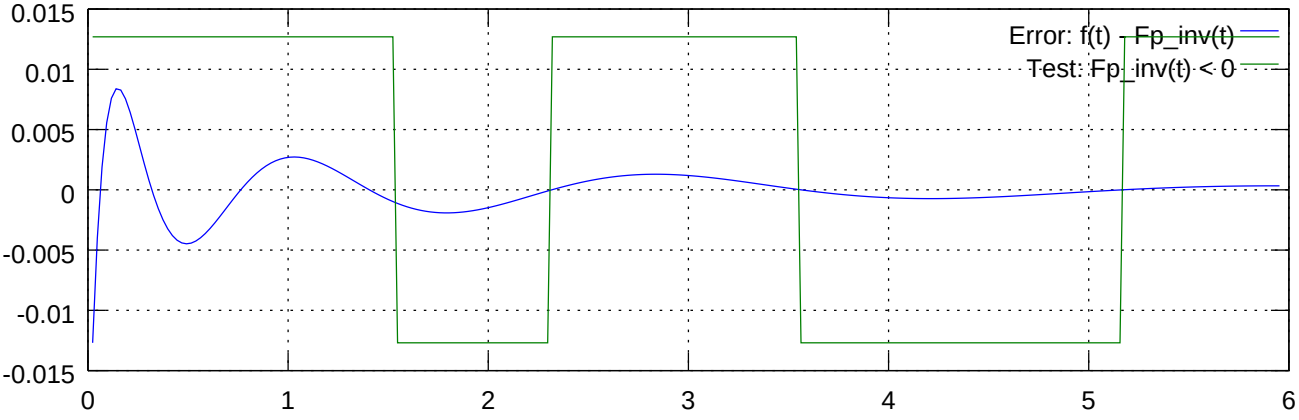


Figure 12: Inversion of $f(t) = e^{-\mu t}$ by Laguerre ($K = 17$) algorithm.

4.2 Algorithms with complex Laplace argument: $p \in \mathbb{C}$

None of the tested inversion algorithms based on real Laplace argument offers satisfying performance. To go further, one now needs to test more general inversion algorithm for which functions to be inverted can be defined in the complex plane.

Let us recall that reservoir models are, most of the time, only known in the Laplace domain for p the Laplace argument being a real number. They have not even always got a closed-form expression. As a way of consequence, algorithms presented hereafter cannot be straightforwardly applied in our context. One makes the assumption that some kind of analytic continuation could be found and applied to the models prior to performing inversion. The existence of such a processing is not discussed in this paper.

Two general Fourier based inversion methods are presented in this section. The first one is directly deduced from expression (4). It belongs to the Dubner-Abate algorithm family [4]. The second one, called Den Iseger's method, is more recent since it appeared in [6].

4.2.1 Fourier algorithm

When considering the inversion relation (8), one straightforwardly can think that direct Fourier based methods are very strong candidates to efficiently invert Laplace transform as stated in [1]. This is especially true when considering the very fast Fourier transform algorithms sample group that is available nowadays. There is no need to know the Laplace function for all the values of $p \in \mathbb{C}$ in the whole complex plane. Indeed, inversion is often based on a simple integration along a line in the complex plane, which is distorted to avoid singularities.

Fourier based methods usually give fairly good inversion results. The inverted function gently converges towards the analytical solution without oscillating much as can be seen in Figure 13. Unfortunately, again, at some point, the inversion result becomes negative. Tweaking the algorithm can allow one to delay this instant but only on a limited range. As a conclusion, even if reservoir models were known in the complex plane, this algorithm would not be suitable for our positive convergence issue.

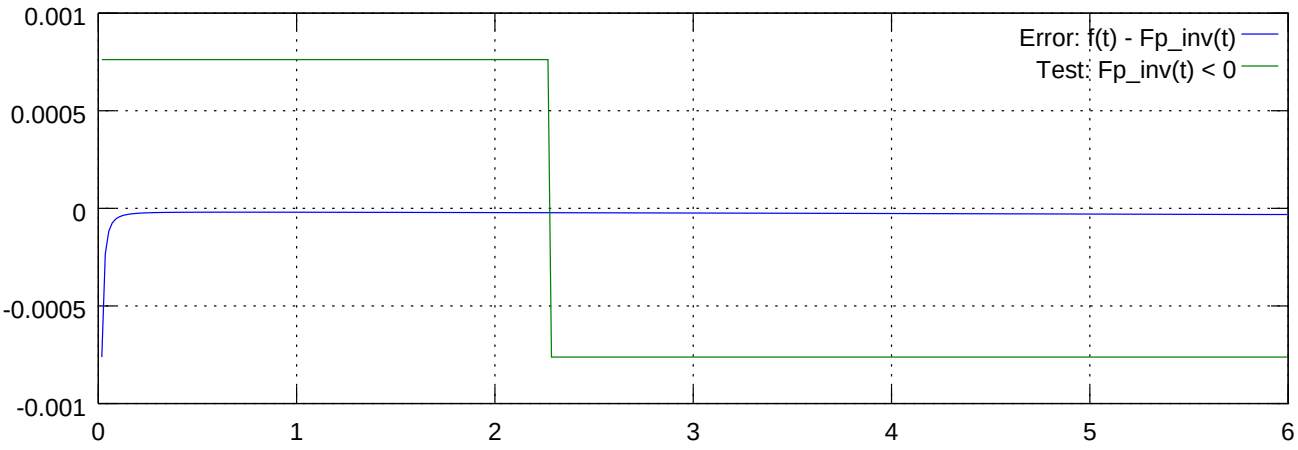


Figure 13: Inversion of $f(t) = e^{-\mu t}$ by Fourier algorithm.

4.2.2 Den Iseger algorithm

The Den Iseger algorithm, described in [6], has recently become very popular in petroleum engineering research, as can be seen in [5]. This algorithm is indeed very accurate. It can also be categorised as a Fourier based algorithm and can only be applied on functions $\hat{F}(p)$ defined in the complex plane with $p \in \mathbb{C}$.

One can see on our tested function in Figure 14 that inversion performance is impressive. Indeed, the reconstruction error is close to machine precision. Furthermore, as can be seen on the sign test curve, the inverted function nicely follows the analytical result and always remains positive. This is precisely the behaviour that has been sought.

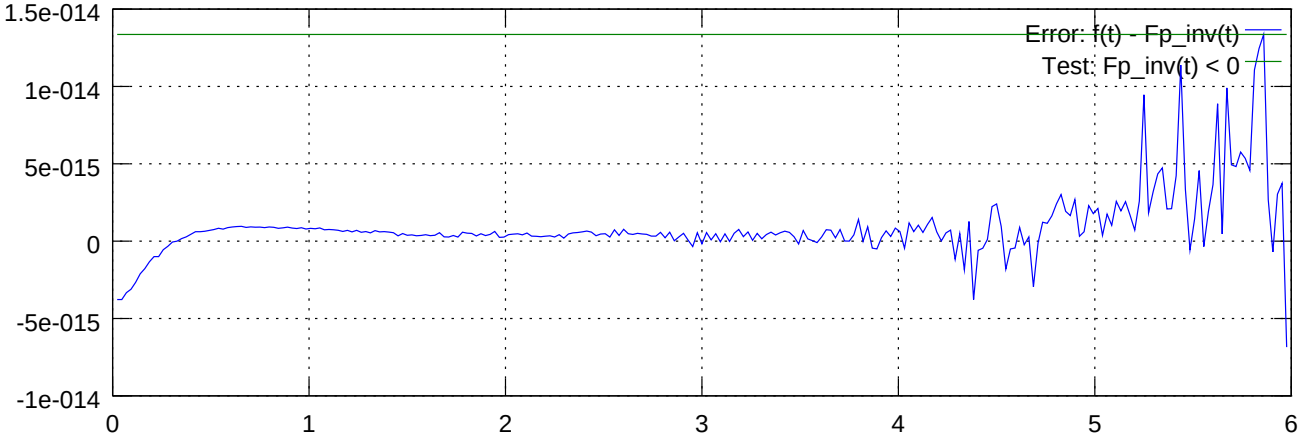


Figure 14: Inversion of $f(t) = e^{-\mu t}$ by Den Iseger algorithm.

Undoubtedly, if Den Iseger algorithm did not require the Laplace argument p to be complex, it would be a serious candidate to solve the issue that is discussed here.

5 Discussion

5.1 A word about algorithms running time

All along this study, we have not focused on algorithms running time. The main reason is that they could not directly be compared. Indeed, no implementation optimisations have been sought, but depending on the algorithm specifications, a variety of shortcuts could be found. Furthermore, several heterogeneous libraries have been used that contributed in either slowing down or speeding up processing times.

The only element that can be compared is the number of times the function $\hat{F}(p)$, defined in the Laplace domain, has been scanned. This measurement is important since in industrial applications this number of calls is at the heart of the processing. Before giving the figures, one first needs to set a few parameters. The typical values given hereafter have be chosen in order to obtain the best performance :

$M = 256$: number of points on which inverted function $f(t)$ is rebuilt

$N_K = 18$: number of terms N or K in the summation

$Q = 2$: Laguerre algorithm number of degrees of freedom

$N_{DFT} = 8M$: number of points used for the Discrete Fourier Transform

$N_L = 8$: Den Iseger algorithm quadrature rule number points

Often, the number of calls to the function $\hat{F}(p)$, defined in the Laplace domain, is proportional to the number of points on which the function is, directly or indirectly, rebuilt in the usual space, i.e. M or N_{DFT}. This is not the case for the polynomials series expansion algorithms.

The number of calls to the function $\hat{F}(p)$ for each algorithm is therefore:

Chebyshev:	$18 = N_K$
Legendre:	$18 = N_K$
Laguerre:	$36 = N_K Q$
Fourier:	$2048 = N_{DFT}$
Gaver-Stehfest:	$4608 = N_K M$
Den Iseger:	$16392 = N_L(N_{DFT} + 1)$

5.2 Tests results

None of the tested algorithm has given fully suitable results. Algorithms based on a real Laplace argument could not properly rebuild an always positive declining exponential function such as $e^{-\mu t}$. Every inversion attempt ended up at some point with negative values. Some kind of tweaking can contribute in improving the inversion performance, but no any universal solution could be found.

When going beyond the main constraint by assuming that the Laplace function to be inverted is defined in the complex plane, one could only find one single algorithm that could invert the function with enough accuracy. The problem is now only shifted, since one has to find a way to build some kind of analytic continuation to set the Laplace functions to be inverted in the complex plane.

Since one can not exhibit an algorithm that straightforwardly solve the issue, one can only give a few hints to mainly correct a posteriori inversion results.

5.2.1 Pre-correction

The Laplace transform is a linear operator. To ensure a more accurate inversion one can add a "correction function", say $\hat{F}_{corr}(p)$, to the function $\hat{F}_{model}(p)$ to be inverted to act on the result. A strategy would be to add a "security" function whose analytical solution is known and that either corrects or minimises known artefacts. In short, instead of calculating :

$$\mathcal{L}^{-1}\left[\hat{F}_{model}(p)\right] = f_{model}(t) \tag{20}$$

one would rather perform :

$$\mathcal{L}^{-1}\left[\hat{F}_{model}(p) + \hat{F}_{corr}(p)\right] = f_{model}(t) + f_{corr}(t) \tag{21}$$

What we call pre-correction can indeed contribute in improving the inversion result. Unfortunately, it may also only delay the artefact or worse, potentially bring instability into a system that was originally stable. Furthermore, there is no general rule to be followed to build up a suitable tailor made correction function.

5.2.2 Post-correction

Another alternative would be to calculate the behaviour of the inverted function towards the infinity and correspondingly perform a corrective post-processing. The Laplace transform offers a powerful theorem that allows one to know the value of the inverted function $f(t)$ towards the infinity when knowing the function $\hat{F}(p)$ in the Laplace domain for $p = 0$:

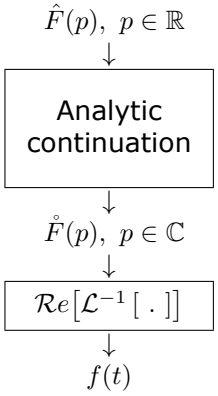
$$\text{final value theorem} \quad \Longleftrightarrow \quad f(+\infty) = \lim_{p \to 0} p\hat{F}(p) \tag{22}$$

After the final value is known, the inversion algorithm can detect the vicinity of this value and adapt its mode to avoid late time inversion artefacts. One can either choose to stop the inversion, or to remove wrong values, or even to extend the reconstructed function with another processing type by asymptotic analysis or algorithm simplification.

5.2.3 Analytic continuation & inversion in the complex plane

Den Iseger algorithm has proven to be the most successful and accurate inversion method among the ones tested here. As already mentioned, it does not entirely satisfy our requirements in the context of petroleum engineering since the function to be inverted needs to be known in the complex plane, i.e. with $p \in \mathbb{C}$.

A solution to fill the gap between the function $\hat{F}(p)$, only known in the Laplace domain for a real Laplace argument $p \in \mathbb{R}$, and this particular inversion algorithm, would be to implement an analytic continuation process. It should be applied on the function $\hat{F}(p)$ and would yield a function $\mathring{F}(p)$ defined in the complex plane for $p \in \mathbb{C}$. This latter function $\mathring{F}(p)$ could eventually be inverted by using a less constrained inversion algorithm such as Den Iseger. The full process would be as follows:



Now, it should be pointed out that analytic continuation is not a trivial task to perform. In short, its principle is to extend definition domain of functions by mapping a real line onto a complex plane. Among the difficulties, one can mention for instance the fact that certain functions with a unique definition in the real space can have several definitions in the complex plane that do not preserve all the properties of the original function, e.g. $\ln(t),\ t \in \mathbb{R} \longleftrightarrow \ln(p),\ p \in \mathbb{C}$. Another difficulty is to define strategies to avoid potential singularities.

Today, nothing warranties that new issues will not be brought to the numerical inversion initial issue by adding this new processing. Furthermore, Den Iseger algorithm has proven to be a good candidate in the current study after assessing performances on only a single test function. A new study should be lead to generalise these first results.

Acknowledgements

We would like to thank Phillip Fair from Shell, Bob Hite from Blue Ridge PTA and Tony Fitzpatrick from Schlumberger for their constructive remarks regarding Laplace transform numerical inversion issues and analytic continuation.

References

[1] Brian Davies and Brian Martin. Numerical inversion of the Laplace transform: a survey and comparison of methods. *Journal of Computational Physics*, 33(1):1–32, October 1979.

[2] Harald Stehfest. Algorithm 368: Numerical inversion of laplace transforms [d5]. *Commun. ACM*, 13:47–49, January 1970.

[3] Harald Stehfest. Remark on algorithm 368: Numerical inversion of laplace transforms. *Commun. ACM*, 13:624–, October 1970.

[4] Harvey Dubner and Joseph Abate. Numerical inversion of laplace transforms by relating them to the finite fourier cosine transform. *J. ACM*, 15(1):115–123, 1968.

[5] Egill Juliusson. *Characterization of Fractured Geothermal Reservoirs based on Production Data*. PhD thesis, Stanford University, Department of Energy Resources Engineering, School of earth sciences, Stanford University, Stanford, California, 2012.

[6] Peter Den Iseger. Numerical transform inversion using gaussian quadrature. *Probab. Eng. Inf. Sci.*, 20(1):1–44, January 2006.

[7] Brian Davies. *Integral Transforms and Their Applications*. Texts in Applied Mathematics. Springer, 2002.

[8] William T. Weeks. Numerical inversion of laplace transforms using laguerre functions. *J. ACM*, 13(3):419–429, July 1966.