NUMERICAL INTEGRATION OF INACCURATELY EVALUATED FUNCTIONS

J. Daněk, J. Pospíšil

NTIS - New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic

Abstract

In this paper we demonstrate the behavior of numerical integration algorithms in situations where the integrands are inaccurately evaluated functions. These problems can occur among others in mathematical finance, e.g. in calibrating option pricing models to real market data. Integrals usually depend on several model parameters and the optimization task consists of large number of integral evaluations with high precision and low computational time requirements. We show that an adaptive quadrature algorithm implemented in the MATLAB's function `integral` fails to meet these requirements, since we can observe an enormous increase in function evaluations, serious precision problems as well as a significant increase of computational time. We demonstrate such behavior on a simplified integrand and we discuss the possible modifications of the integration process to solve the raised issues.

1 Introduction

In mathematical finance, a process of calibrating option pricing models to real market data is usually formulated as an optimization problem of nonlinear least squares (Mrázek et al., 2014, 2015). In this paper we focus on the continuous-time stochastic volatility (SV) models whose pricing formulas are often available in the semi-closed form - they involve an infinite-domain integral that has to be calculated numerically. Although there exist many SV models that could be studied independently, we take advantage of a recent results by Baustian et al. (2015) who introduced a unifying approach to stochastic volatility jump-diffusion (SVJD) models covering among others the popular SV model by Heston (1993), SVJD model by Bates (1996) as well as the recently proposed asymptotically fractional SVJD model (Pospíšil and Sobotka, 2015). Numerical problems discussed in this paper therefore occur not only in the unifying formula, but also in all original models that are widely used in practice.

Structure of the paper is as follows. In Section 2 we introduce the problem of numerical integration of inaccurately evaluated functions that occur in pricing formulas for stochastic volatility models. In Section 3 we perform the numerical analysis. At first we explain the numerical behavior using a simplified problem and then perform a comparison of different quadratures for the original problem. We conclude in Section 4.

2 Problem description

Let us consider a unifying formula (Baustian et al., 2015) for the Bates (1996) SVJD model.

$$V(S,v,\tau) = S - Ke^{-r\tau}\frac{1}{2\pi}\int_{-\infty+ik_i}^{\infty+ik_i} e^{-ik\tilde{X}}\frac{\hat{H}(k,v,\tau)}{k^2-ik}\exp\{\lambda(\hat{\varphi}(-k)-1)\tau\}dk, \tag{1}$$

where $k_i = 1/2$, i is the imaginary unit and

$$\begin{aligned}
\tilde{X} &= \ln\frac{S}{K} + (r - \lambda\beta)\tau, \\
\beta &= \exp\left\{\mu_J + \frac{1}{2}\sigma_J^2\right\} - 1, \\
\hat{\varphi}(u) &= \exp\left\{i\mu_J u - \frac{1}{2}\sigma_J^2 u^2\right\}, \\
\hat{H}(k, v, \tau) &= \exp\left(\frac{2\kappa\theta}{\sigma^2}\left[q\,g - \ln\left(\frac{1 - he^{-\xi q}}{1 - h}\right)\right] + vg\left(\frac{1 - e^{-\xi q}}{1 - he^{-\xi q}}\right)\right), \\
g &= \frac{b - \xi}{2}, \quad h = \frac{b - \xi}{b + \xi}, \quad q = \frac{\sigma^2\tau}{2}, \\
\xi &= \sqrt{b^2 + \frac{4(k^2 - ik)}{\sigma^2}}, \\
b &= \frac{2}{\sigma^2}\Big(ik\rho\sigma + \kappa\Big).
\end{aligned}$$

When calculating numerically the above integral, some values of the model parameters can affect the accuracy of the integrand when using the standard floating point arithmetic (`double`), where significant digits can get lost. Let us consider the following data and parameter values:

$S = 6721.8$, $v = 0.97114$, $\tau = 0.120548$, $K = 6250$, $r = 0.009$, $\lambda = 11.70271$, $\mu_J = -6.65876$, $\sigma_J = 1.0069$, $\theta = 0.95333$, $\kappa = 17.6751$, $\rho = -0.86219$ and $\sigma = 0.000028$.

If we depict the integrand over the interval $[0, 10]$, for the first sight we observe a "nice" smooth-looking function (see Figure 1). However, when we use a finer discretization and zoom in, a neighborhood of the point 3 does not look smooth anymore (see the blue curve in Figure 2).

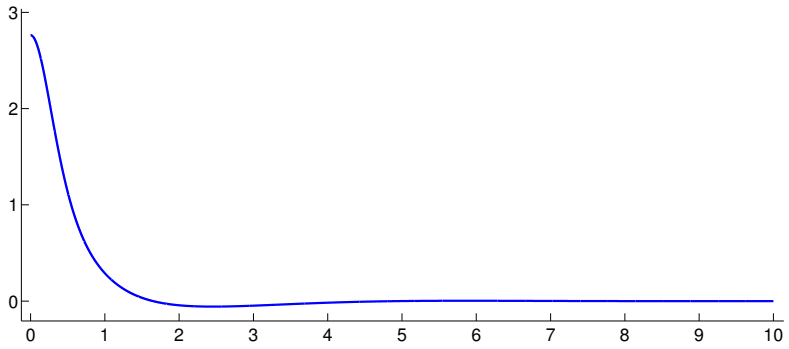

Figure 1: Global view of integrated function - wider horizon.

This effect is caused by an insufficient precision arithmetic used and in the graph of the integrand we can observe jumps. Size and frequency of the jumps depends also on the value of the independent variable k, i.e. jump sizes decrease with k, but the frequency increases.

We can use the variable-precision arithmetic (`vpa`) and specify the number of significant digits to obtain a better precision when evaluating the integrand. In Figure 2, the red curve represents the same integrand evaluated using precision to 32 significant digits. It is depicted at the same discrete points and we can see that it does not contain jumps anymore.

Let us now try to integrate the considered function in the given region $[0, 10]$. If we use the standard floating-point arithmetic that is used by the MATLAB's function `integral`, there is a certain chance to get into troubles. Using the default tolerance values `'AbsTol'=1e-6` and `'RelTol'=1e-6`, the function `integral` returns the value 0.776588380039885 in ca 0.73 [sec]

(measured by functions `tic` and `toc` on a reference computer: dual core Intel i7, 2.9GHz, 64bit Windows 10, MATLAB R2015a). This calculation requires 150 function evaluations (fevals) used by the adaptive algorithm. Although all these values seem to be acceptable, a slight change in one of the parameters can cause significantly different behavior. If we change the value of σ from the original value 0.000028 to a new value 0.000022 and we keep all the other values unchanged, we get the value of the integral 0.776583174460231, it is calculated in ca 3.82 [sec] and the number of fevals during the adaptive algorithm grows to 127680.

We could decrease the tolerance (`'AbsTol'` and `'RelTol'`) requirements to overcome this dramatic increase in number of fevals, however, for optimization problems that occur during the calibration process, high precision results and low computation times are requisite.

To explain the different behavior of function `integral` for almost identical input data, we have to realize that a requested tolerance is tested in each iteration of the adaptive quadrature algorithm and it is influenced by the imprecision jumps in the integrand. If we apply the integration procedure to function that is evaluated using the variable-precision arithmetic (`vpa`), then in both above cases the integral routine requires 180 function evaluations.

In Table 1 we can observe a computation time increase when using `vpa` compared to `double`, which is caused by rather complicated formula for the integrand. On the other hand, integral in the problematic case ($\sigma = .000022$) is computed in a comparable time. As for the accuracy, only the result using `vpa` can be considered trustworthy.

Table 1: Comparison for two different input parameters.

		value of integral	time [sec]	fevals
$\sigma = .000028$	`double`	0.776588380039885	0.73	150
	`vpa`	0.776585856534337	4.63	180
$\sigma = .000022$	`double`	0.776583174460231	3.82	127680
	`vpa`	0.776585856549349	4.49	180

In formula (1), the integral should be calculated over the infinite domain $[0, \infty)$, for increasing argument the integrand goes to zero rather quickly. In practice we consider only finite length domains. In Figure 3 we can see the integrand over the interval $[0, 100]$.

In Tables 2 and 3 we can compare results for different integration domains for both cases $\sigma = .000028$ and $\sigma = .000022$ respectively. Whereas in the former case, the behavior (time and number of fevals) is rather unexpectable, in the latter case we can observe that with the higher upper integration bound we get the higher computation time and more fevals is needed. In both cases, when using `vpa`, fevals increase only from 180 to 270 whereas in `double` case they blow up to 112350 or 526620 respectively. Values of the integral evaluated for two different values of

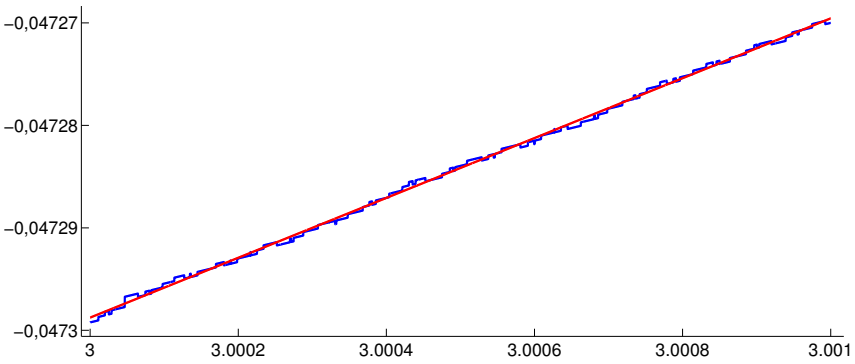

Figure 2: Detailed view of the integrand.

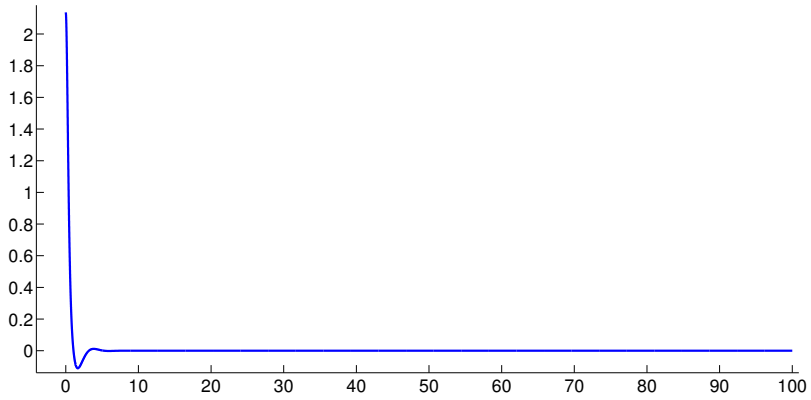

Figure 3: Global view to integrated function.

σ differ in the `vpa` case only over the interval $[0, 10]$ (the first row in both tables) as described above.

As for the computation times, for upper integration bounds 30 and above, calculations using `vpa` are faster (and more precise) than those using `double`. This is caused by extremely large number of fevals in the latter case while in `vpa` case it remains relatively low. As for the integration values, standard `double` precision is rather low and the values of the integral are for both values of σ the same provided that the upper integration bound is grater than 20. The value obtained by `vpa` converges to the exact value faster.

Table 2: Comparison for different integration domains, case $\sigma = .000028$.

	`double`			`vpa`		
	value of integral	time [sec]	fevals	value of integral	time [sec]	fevals
$[0, 10]$	0.776588380039885	0.73	150	0.776585856534337	4.63	180
$[0, 20]$	0.776580751834288	0.04	180	0.776578226075433	5.53	210
$[0, 30]$	0.776578160122915	1.78	55800	0.776578226075448	5.48	210
$[0, 40]$	0.776580762365733	0.04	210	0.776578226075447	6.31	240
$[0, 50]$	0.776578273605539	0.71	22680	0.776578226075473	6.40	240
$[0, 60]$	0.776578215111363	2.72	88170	0.776578226075448	6.23	240
$[0, 70]$	0.776578753464731	0.05	540	0.776578226075336	6.30	240
$[0, 80]$	0.776580740711720	0.05	270	0.776578226075448	7.03	270
$[0, 90]$	0.776578061270396	3.44	112350	0.776578226075419	7.06	270
$[0, 100]$	0.776578145241924	1.74	54360	0.776578226075483	7.02	270

3 Numerical analysis

At first let us explain the behavior described in the previous section using a simplified problem that we can obtain by a successive simplification of the original integrand. Let us consider the following function that has similar properties:

```
function result=f_double(x);
  sigma  =  1e-9;
  a      = x + 1000./sigma;
  b      = (a.^2+(x.^2)./sigma).^.5;
  result = a.^2-b.^2;
end
```

We can analytically simplify this formula even further to:

Table 3: Comparison for different integration domains, case $\sigma = .000022$.

	double			vpa		
	value of integral	time [sec]	fevals	value of integral	time [sec]	fevals
$[0, 10]$	0.776583174460231	3.82	127680	0.776585856549349	4.49	180
$[0, 20]$	0.776575614352789	4.87	165180	0.776578226092740	5.20	210
$[0, 30]$	0.776575576765266	6.98	236760	0.776578226092755	5.18	210
$[0, 40]$	0.776575531502838	5.94	198930	0.776578226092754	5.92	240
$[0, 50]$	0.776575627836667	11.13	379110	0.776578226092780	5.94	240
$[0, 60]$	0.776575609084378	9.55	324090	0.776578226092755	5.89	240
$[0, 70]$	0.776575652981054	8.99	308220	0.776578226092643	5.91	240
$[0, 80]$	0.776575593708892	15.37	526620	0.776578226092756	6.65	270
$[0, 90]$	0.776575594664419	15.04	492270	0.776578226092727	7.25	270
$[0, 100]$	0.776575640834488	14.79	505230	0.776578226092791	6.65	270

```
function result=f_exact(x);
  sigma  =  1e-9;
  result = -(x.^2)./sigma;
end
```

Our function implemented using `vpa` looks as follows:

```
function result=f_vpa(x,dig);
  old=digits(dig);
  sigma  = vpa(1e-9);
  a      = vpa(x + 1000./sigma);
  b      = vpa((a.^2+(x.^2)./sigma).^.5);
  result = vpa(a.^2-b.^2);
  result = double(result);
  digits(old);
end
```

In Figures 4-7 there are graphs of studied functions depicted over the interval $[0, 10]$ or in a neighborhood of number 3. Function `f_double(x)` is blue and functions `f_exact(x)` and `f_vpa(x)` are both red.

Let us now integrate functions `f_double(x)` and `f_vpa(x)` over the interval $[0, 10]$ using function `integral` with `'AbsTol'=1e-6` and `'RelTol'=1e-6`.

To calculate the integral of `f_double(x)` we needed 7.64 [sec], 492300 fevals and the inaccurate result was $-3.333305894598757e + 11$. For function `f_vpa(x)` we needed 2.83 [sec], 150 fevals and the result $-3.333333333333333e+11$ was obtained accurately. In the first case, the function integral may produce the following warning indicating that there might be something wrong with the calculation.

```
Warning: Reached the limit on the maximum number of intervals
in use. Approximate bound on error is 6.8e+07. The integral
may not exist, or it may be difficult to approximate numerically
to the requested accuracy.
> In funfun\private\integralCalc>iterateArrayValued at 282
  In funfun\private\integralCalc>vadapt at 130
  In funfun\private\integralCalc at 75
  In integral at 88
```

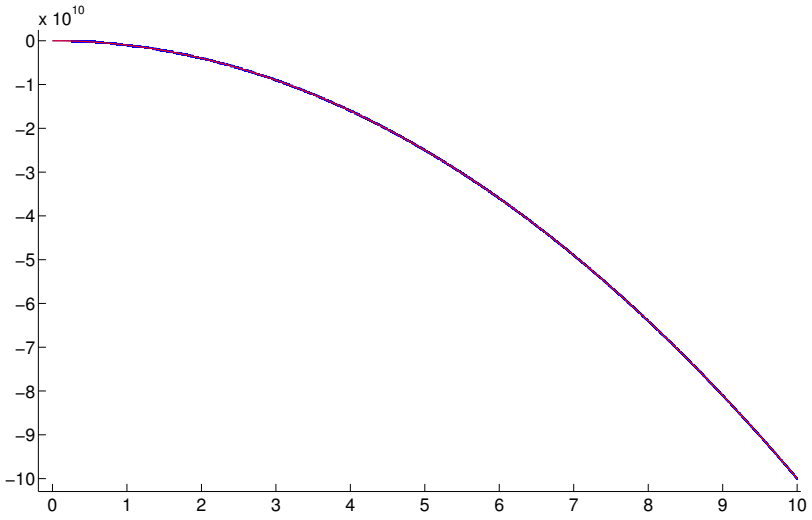

Figure 4: Global view to `f_double(x)` (blue) and `f_vpa(x)` (red).

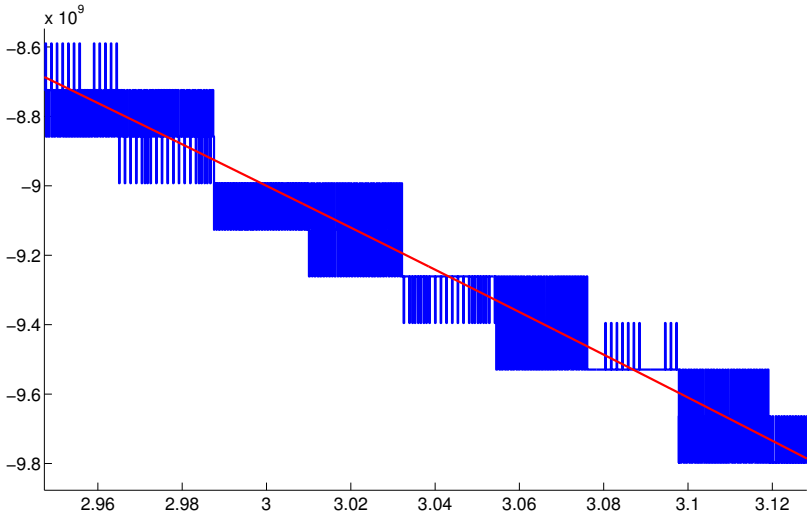

Figure 5: Detailed view to `f_double(x)` (blue) and `f_vpa(x)` (red).

The influence of the lower relative tolerance is such that for example for `'RelTol'=1e-3` fevals will be in both cases only 150, however the result $-3.332756082277576e{+}11$, for `f_double(x)` will be even less accurate whereas for `f_vpa(x)` we get again $-3.333333333333333e+11$.

Let us now have a closer look (see Table 4) to results obtained at different integration domains, in particular over the intervals $[0,1]$, $[0,2]$ up-to $[0,10]$. For `f_double(x)` and an upper bound 3 and higher MATLAB returns also the same type of warning message. We can again observe a remarkable differences in number of fevals and in obtained accuracy. For `f_vpa(x)` the results are accurate to requested number of significant digits. Calculation times are almost identical in both cases, times are smaller for `f_vpa(x)`. Furthermore, for intervals $[0,1]$ and $[0,2]$ the computation times and number of fevals were bigger.

So far we have used only the function `integral` (available in MATLAB since 2012) that implements the adaptive Gauss7Kronrod15 quadrature (Shampine, 2008, 2010). MATLAB offers also several other quadratures although they remain in the distribution as unsupported. In Table 5 we can compare the results of integration of both functions `f_double(x)` and `f_vpa(x)` over the interval $[0,10]$ using functions `integral`, `quad` (adaptive Simpson's rule), `quadl` (adaptive

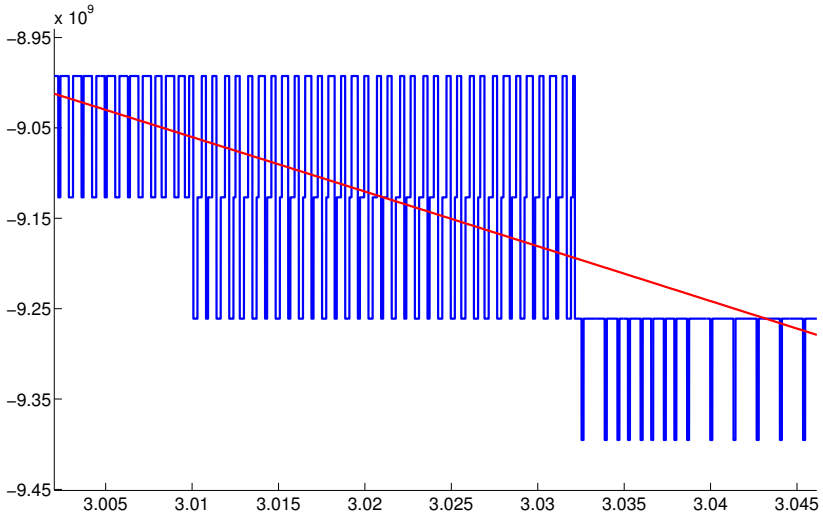

Figure 6: Detailed view to `f_double(x)` (blue) and `f_vpa(x)` (red).

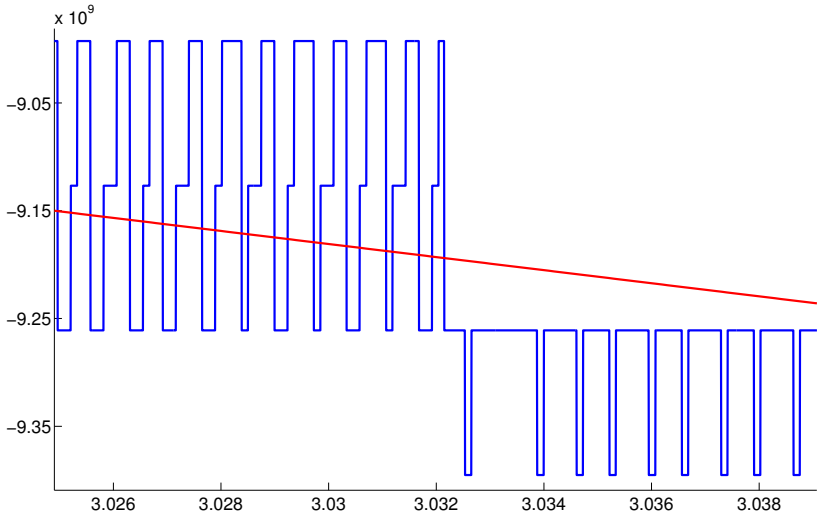

Figure 7: Detailed view to `f_double(x)` (blue) and `f_vpa(x)` (red).

Table 4: INTEGRATION OF `F_DOUBLE(X)` AND `F_VPA(X)` OVER DIFFERENT DOMAINS.

	`f_double(x)`			`f_vpa(x)`		
	value of integral	time	fevals	value of integral	time	fevals
$[0, 1]$	-3.288591668013657e+08	18.16	1165980	-3.333333333333333e+08	2.88	150
$[0, 2]$	-2.662111340106951e+09	32.68	2172660	-2.666666666666667e+09	2.79	150
$[0, 3]$	-8.996295897959026e+09	9.51	617760	-8.999999999999998e+09	2.84	150
$[0, 4]$	-2.132780923861364e+10	6.63	422280	-2.133333333333333e+10	3.26	150
$[0, 5]$	-4.166115942073217e+10	6.91	449250	-4.166666666666666e+10	2.95	150
$[0, 6]$	-7.200721626636041e+10	6.40	415740	-7.199999999999999e+10	2.73	150
$[0, 7]$	-1.143302905838337e+11	7.30	472020	-1.143333333333333e+11	2.75	150
$[0, 8]$	-1.706596836888415e+11	7.63	480510	-1.706666666666667e+11	2.71	150
$[0, 9]$	-2.429973311534886e+11	7.53	484920	-2.429999999999999e+11	2.84	150
$[0, 10]$	-3.333305894598757e+11	7.64	492300	-3.333333333333333e+11	2.83	150

Gauss-Lobatto quadrature, Gander and Gautschi (2000)) and `trapz` (simple trapezoidal rule).

Table 5: INTEGRATION OF `F_DOUBLE(X)` AND `F_VPA(X)` USING DIFFERENT METHODS.

	`f_double(x)`		`f_vpa(x)`	
	value of integral	time	value of integral	time
`integral`	-3.333305894598757e+11	7.64	-3.333333333333333e+11	2.83
`quad`	-3.330331572927783e+11	0.16	-3.333333333333333e+11	0.10
`quadl`	-3.388795564650947e+11	0.09	-3.333333333333334e+11	0.08
`trapz` step=1e-2	-3.333619397427199e+11	0.0007	-3.333335000000002e+11	1.83
step=1e-3	-3.333298617057282e+11	0.0018	-3.333333350000001e+11	3.52
step=1e-4	-3.333293382565890e+11	0.01	-3.333333333500004e+11	33.39
step=1e-5	-3.333285611359462e+11	0.06	-3.333333333334993e+11	355.83

Both `quad` and `quadl` when integrating the function `f_double` returns also the following warning message

```
Warning: Maximum function count exceeded; singularity likely.
> In quad at 98
```

or

```
Warning: Maximum function count exceeded; singularity likely.
> In quadl at 98
```

respectively.

From the comparison we can read that the fastest was `trapz` for inaccurate function `f_double(x)`, the precision is however far from being satisfactory. Using `trapz` for accurate function `f_vpa(x)` is on the other hand time consuming. Using quadratures `quad` and `quadl` for accurately evaluated function `f_vpa(x)` keeps the computation time low with attaining the required precision.

Let us now perform a similar comparison for the original problem (calculating the integral in formula (1)). In Table 6 we extend the results from Table 3 (function `integral` for `double` and `vpa`) by results obtained by `quad` and `quadl` over different integration domains. We can observe that both `quad` and `quadl` are faster than `integral`. Let the values obtained by `integral` (`vpa`) be reference values. From Table 6 we can read that `quadl` is sufficiently precise (see the red values), but several times (ca $4-5$ times) faster than `integral` (`vpa`).

4 Conclusion

In this paper we aimed at numerical evaluation of the integral from formula (1) for the Bates stochastic volatility jump-diffusion model. There are several data and model parameters with specified simple (lower and upper) bounds and during the calibration process (fitting model parameters to real market data) all values in these bounds can be reached. In this optimization task, the integral is evaluated many times (ca in orders of thousands, each time with different parameter values) and if there are a lot of options (there are usually ca 100-200 options per asset per day in one calibration task) the value of the integral must be calculated sufficiently accurately (with relative error ca `1e-9`).

In several calibration tasks we performed on real market data, we noticed that particular parameter values extremely slowed down the numerical integration. A deeper analysis showed that the numerically integrand (that is supposed to be regular and smooth) is evaluated inaccurately (with discontinuities) due to insufficient arithmetic precision (`double`). It is worth to mention that this misbehavior is rather rare and in many combinations of admissible parameter values the standard (`double`) precision is sufficient for evaluating the integral and a better (`vpa`)

Table 6: Integration of the original function over different domains and using different quadratures.

	`integral (double)`		`integral (vpa)`		`quad (vpa)`		`quadl (vpa)`	
	value of integral difference towards integral (vpa)	time [sec]	value of integral	time [sec]	value of integral difference towards integral (vpa)	time	value of integral difference towards integral (vpa)	time [sec]
$[0, 10]$	0.776583174460231 -0.2682**e-5**	3.82	0.776585856549349 **0**	4.49	0.776585839198780 -0.1735**e-7**	1.31	0.776585856546556 -0.2792**e-11**	1.15
$[0, 20]$	0.776575614352789 -0.2611**e-5**	4.87	0.776578226092740 **0**	5.20	0.776579189869512 0.9637**e-6**	1.37	0.776578225081295 -0.1011**e-8**	1.13
$[0, 30]$	0.776575576765266 -0.2649**e-5**	6.98	0.776578226092755 **0**	5.18	0.776578018942537 -0.2071**e-6**	1.37	0.776578226436952 0.3441**e-9**	1.32
$[0, 40]$	0.776575531502838 -0.2694**e-5**	5.94	0.776578226092754 **0**	5.92	0.776573061882257 -0.5164**e-5**	1.44	0.776578266484348 0.4039**e-7**	1.32
$[0, 50]$	0.776575627836667 -0.2598**e-5**	11.13	0.776578226092780 **0**	5.94	0.776578424336860 0.1982**e-6**	1.37	0.776578226906582 0.8138**e-9**	1.33
$[0, 60]$	0.776575609084378 -0.2617**e-5**	9.55	0.776578226092755 **0**	5.89	0.776577988217763 -0.2378**e-6**	1.39	0.776578226518933 0.4261**e-9**	1.34
$[0, 70]$	0.776575652981054 -0.2573**e-5**	8.99	0.776578226092643 **0**	5.91	0.776579991411628 0.1765**e-5**	1.55	0.776578225801549 -0.2910**e-9**	1.36
$[0, 80]$	0.776575593708892 -0.2632**e-5**	15.37	0.776578226092756 **0**	6.65	0.776575596029267 -0.2630**e-5**	1.50	0.776578226144205 0.5144**e-10**	1.52
$[0, 90]$	0.776575594664419 -0.2631**e-5**	15.04	0.776578226092727 **0**	7.25	0.776578196059217 -0.3003**e-7**	1.48	0.776578226113687 0.2096**e-10**	1.53
$[0, 100]$	0.776575640834488 -0.2585**e-5**	14.79	0.776578226092791 **0**	6.65	0.776578411330985 0.1852**e-6**	1.52	0.776578226051067 -0.4172**e-10**	1.60

precision is not needed, since the values obtained by the adaptive Gauss7Kronrod15 quadrature implemented in function `integral` is fast and sufficiently accurate.

On the other hand, for some parameter values, the standard (`double`) accuracy is insufficient and causes serious troubles in the numerical integration and it is necessary to use a better (`vpa`) arithmetic precision. Since the integrand is a rather complicated function, its evaluation in `vpa` is time consuming. We have showed that having an accurately evaluated functions can lower significantly the number of function evaluations in the adaptive numerical quadratures and consequently can lower computation times and especially we get more accurate results of the integration.

Thorough testing showed that sufficiently accurate results can be obtained significantly faster by the Gauss-Lobatto quadrature implemented in function `quadl` (and using the `vpa` for the integrand evaluation of course).

Further study of different numerical quadratures and modified techniques for adaptive integration is requisite and it is now an ongoing research.

Acknowledgement

This work was supported by the GACR Grant 14-11559S Analysis of Fractional Stochastic Volatility Models and their Grid Implementation. Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144.

References

D. S. Bates. Jumps and stochastic volatility: exchange rate processes implicit in deutsche mark options. *Review of Financial Studies*, 9(1):69–107, 1996.

F. Baustian, M. Mrázek, J. Pospíšil, and T. Sobotka. Unifying approach to several stochastic volatility models with jumps. Manuscript submitted for publication, 2015.

W. Gander and W. Gautschi. Adaptive quadrature—revisited. *BIT*, 40(1):84–101, 2000. ISSN 0006-3835.

S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(2):327–343, 1993.

M. Mrázek, J. Pospíšil, and T. Sobotka. On optimization techniques for calibration of stochastic volatility models. In *Applied Numerical Mathematics and Scientific Computation*, pages 34–40, Athens, 2014.

M. Mrázek, J. Pospíšil, and T. Sobotka. On calibration of stochastic and fractional stochastic volatility models. Manuscript submitted for publication, 2015.

J. Pospíšil and T. Sobotka. Market calibration under a long memory stochastic volatility model. Manuscript submitted for publication, 2015.

L. F. Shampine. Vectorized adaptive quadrature in Matlab. *J. Comput. Appl. Math.*, 211(2): 131–140, 2008. ISSN 0377-0427.

L. F. Shampine. Weighted quadrature by change of variable. *Neural Parallel Sci. Comput.*, 18 (2):195–206, 2010. ISSN 1061-5369.

Doc. Ing. Josef Daněk, Ph.D.
NTIS - New Technologies for the Information Society
Faculty of Applied Sciences
University of West Bohemia
Technická 8
306 14 Plzeň
Czech Republic
Email: danek@kma.zcu.cz

Ing. Jan Pospíšil, Ph.D.
NTIS - New Technologies for the Information Society
Faculty of Applied Sciences
University of West Bohemia
Technická 8
306 14 Plzeň
Czech Republic
Email: honik@kma.zcu.cz