# Variable precision arithmetic in option pricing formulas

Josef Daněk*, Jan Pospíšil*

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

Moderné nástroje pre finančnú analýzu a modelovanie 2017
Národná banka Slovenska, Bratislava

8. června 2017


# Outline

# Stochastic volatility jump diffusion models

## (SVJD) models

We consider a general SVJD model which covers several kinds of stochastic volatility processes and also different types of jumps

$$\begin{aligned} dS_t &= (r - \lambda\beta)S_t dt + \sqrt{v_t} S_t dW_t^S + S_{t-} dQ_t, \\ dv_t &= p(v_t)dt + q(v_t)dW_t^v, \\ dW_t^S dW_t^v &= \rho\, dt, \end{aligned}$$

where $p, q \in C^\infty(0,\infty)$ are general coefficient functions, $r$ is the interest rate, $\rho$ is the correlation of Wiener processes $W_t^S$ and $W_t^v$, parameters $\lambda$ and $\beta$ correspond to a specific jump process $Q_t$, see below.

F. Baustian, M. Mrázek, J. Pospíšil and T. Sobotka (2017). *Unifying pricing formula for several stochastic volatility models with jumps*, Appl. Stoch. Models Bus. doi: 10.1002/asmb.2248. To appear (online first 04/2017).

# Stochastic volatility jump diffusion models

## General model with particular examples

$$
\begin{aligned}
dS_t &= (r - \lambda\beta)S_t dt + \sqrt{v_t} S_t dW_t^S + S_{t-} dQ_t, \\
dv_t &= p(v_t)dt + q(v_t)dW_t^v, \\
dW_t^S dW_t^v &= \rho\, dt.
\end{aligned}
$$

| model | $p(v)$ | $q(v)$ |
|---|---|---|
| Heston/Bates | $\kappa(\theta - v)$ | $\sigma\sqrt{v}$ |
| 3/2 model* | $\omega v - \tilde{\theta} v^2$ | $\xi v^{\frac{3}{2}}$ |
| Geometric BM | $\alpha v$ | $\xi v$ |
| Approximative fractional SVJD** | $(H - 1/2)\psi_t \sigma\sqrt{v} + \kappa(\theta - v)$ | $\varepsilon^{H-1/2}\sigma\sqrt{v}$ |

$^*\tilde{\theta} = -\frac{1}{2}\xi^2 + (1-\gamma)\rho\xi + \sqrt{(\theta + \frac{1}{2}\xi^2)^2 - \gamma(1-\gamma)\xi^2}$,
$^{**}\psi_t = \int_0^t (t - s + \varepsilon)^{H-3/2} dW_s^{\psi}$.

The jump process $Q_t$ is a compound Poisson process.

Price of a European call option with strike price $K$ and time to maturity $\tau$:

$$V = S - Ke^{-r\tau}\frac{1}{\pi}\int_0^{+\infty+i/2} \underbrace{e^{-ikX}\frac{\hat{F}(k,v,\tau)}{k^2-ik}\phi(-k)}_{f(k)}\,dk,$$

with $X = \ln\frac{S}{K} + r\tau$, fundamental transform $\hat{F}(k,v,\tau) = \exp(C(k,\tau) + D(k,\tau)v)$ and

$$C(k,\tau) = \kappa\theta\underbrace{\left(Y\tau - \frac{2}{B^2}\right)}_{C_1(k,\tau)}\underbrace{\ln\left(\frac{1-ge^{-d\tau}}{1-g}\right)}_{C_2(k,\tau)}, \quad D(k,\tau) = Y\frac{1-e^{-d\tau}}{1-ge^{-d\tau}}, \quad Y = -\frac{k^2-ik}{b+d},$$

$$g = \frac{b-d}{b+d}, \quad d = \sqrt{b^2 + B^2(k^2-ik)}, \quad b = \kappa + ik\rho B, \quad B = \varepsilon^{H-1/2}\sigma,$$

$$\phi(k) = \exp\left\{-i\lambda\beta k\,\tau + \lambda\tau\left[\hat{\varphi}(k) - 1\right]\right\}, \quad \hat{\varphi}(k) = \exp\left\{i\mu_J k - \frac{1}{2}\sigma_J^2 k^2\right\},$$

$$\beta = \exp\{\mu_J + \sigma_J^2/2\} - 1.$$

Typical simple lower and upper real-valued bounds (LB and UB) for AFSVJD model parameters $\chi = (v_0, \kappa, \theta, \sigma, \rho, \lambda, \mu_J, \sigma_J, H)$

| | $v_0$ | $\kappa$ | $\theta$ | $\sigma$ | $\rho$ | $\lambda$ | $\mu_J$ | $\sigma_J$ | $H$ |
|---|---|---|---|---|---|---|---|---|---|
| LB: | 0 | 0 | 0 | 0 | -1 | 0 | -10 | 0 | 0.5 |
| UB: | 1 | 100 | 1 | 4 | 1 | 100 | 5 | 4 | 1.0 |

Ranges that are typical for many option market data sets (options on major indices like DAX, FTSE 100, Nikkei 225 and S&P 500 that occurred at the market in the last two years):

price of the underlying asset (usually in dollars): $0 < S \leq S_{\max} = 30\,000$,

time to maturity (in years): $0 < \tau \leq \tau_{\max} = 5$,

strike price (in the same currency as $S$): $0 < K \leq K_{\max} = 3S_{\max}$,

interest rate (positive): $0 < r \leq r_{\max} = 0.05$.

We will denote the vector of market data by $\psi = (\tau, K, r, S)$.

# Pitfalls of numerical integration

For some model parameters, we can observe (especially for adaptive quadrature algorithms):

- an enormous increase in function evaluations,
- serious precision problems as well as
- a significant increase in computational time.

Problems are caused by inaccurately evaluated integrands:

- all models (including Heston) affected,
- especially sensitive to the value of volatility of volatility $\sigma$,
- standard `double` vs. variable precision arithmetic (`vpa`).

J. Daněk and J. Pospíšil (2015). *Numerical integration of inaccurately evaluated functions*. In Technical Computing Prague 2015. Prague: Czech Technical University, p. 1-11.

J. Daněk and J. Pospíšil (2017). *Numerical aspects of integration in semi-closed option pricing formulas for stochastic volatility jump-diffusion models*. Manuscript under review (submitted 01/2017).

# Pitfalls of numerical integration

## AFSVJD example: global view to integrated function

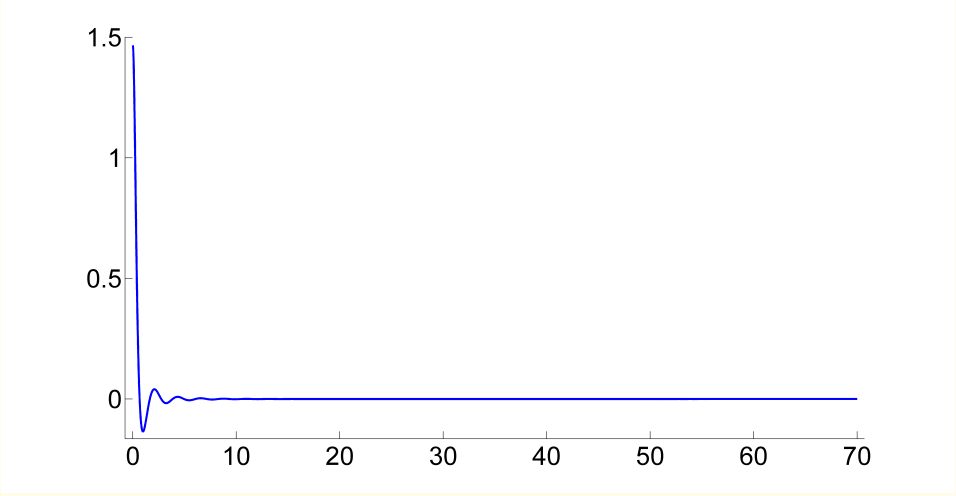



Figure: Parameters: $v_0 = 0.1$, $\kappa = 2.1$, $\theta = 0.4$, $\sigma = 0.002$, $\rho = -0.3$, $\lambda = 25$, $\mu_J = -4$, $\sigma_J = 1.7$, $H = 0.8$, for $S_0 = 6721.8$, $K = 6250$, $\tau = 0.120548$, $r = 0.009$.

# Pitfalls of numerical integration

## AFSVJD example: detailed zoom of integrated function

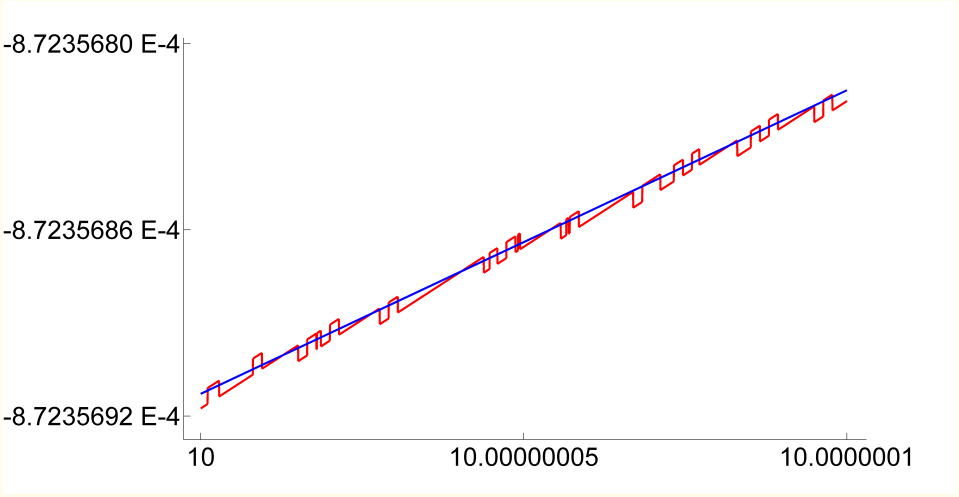
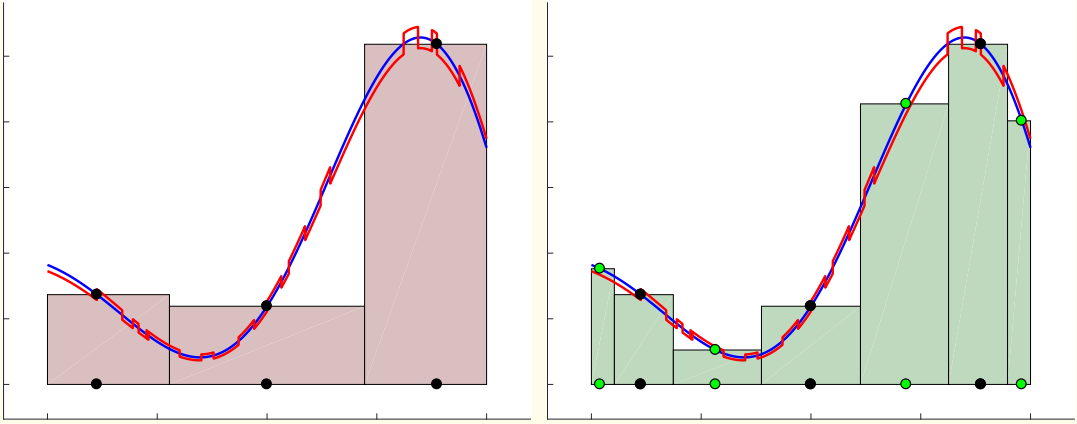



Figure: Same parameters as before. Inaccurately enumerated values in standard `double` (red) and in `vpa` (blue) evaluated with 40 significant digits.

# Pitfalls of numerical integration

## How adaptive quadratures work

- quadrature rules applied on adaptively refined subintervals of the integration domain (error estimate compared to a given tolerance)
- for example in GaussKronrod(3,7) two quadratures are compared:



- MATLAB `intergal` uses GaussKronrod(7,15)

# Pitfalls of numerical integration

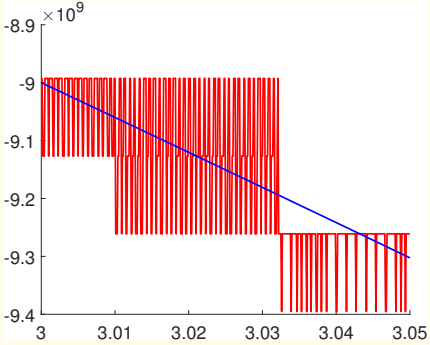
## How `vpaintegral` works

- introduced in MATLAB R2016b as "numerical integration using vpa",
- uses symbolic operations to `simplify` the integrand first

```
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
```

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

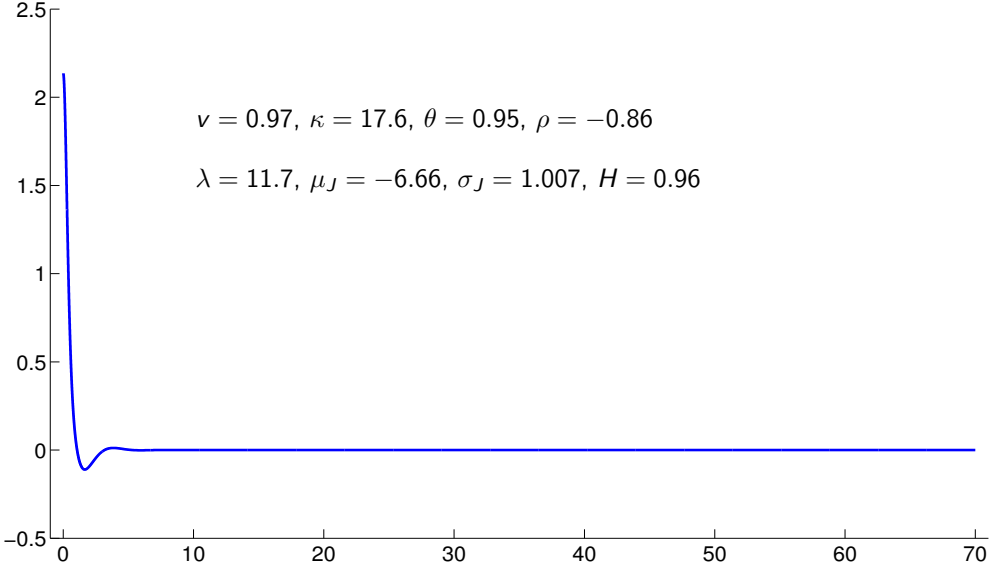
$$\int_0^{10} \left(-x^2/10^{-9}\right) dx = \frac{1}{3}10^{12}$$




- exact value: `-3.333333333333333e+11`
- error between `integral` and `vpaintergal`: `2.743873457580566e+06`
- promising idea, now works for simple integrands (including e.g. Bessel functions), but still has some severe problems

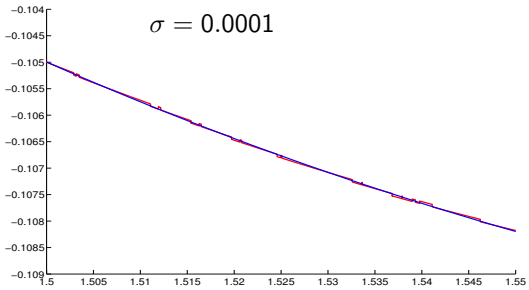
# Inaccurately evaluated integrand

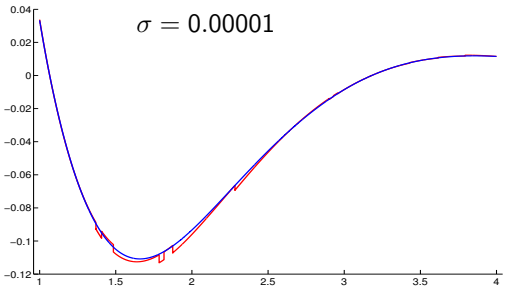
## Test case 1

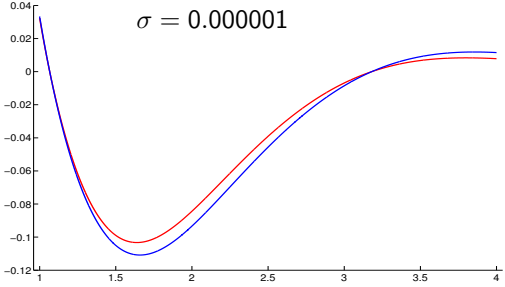


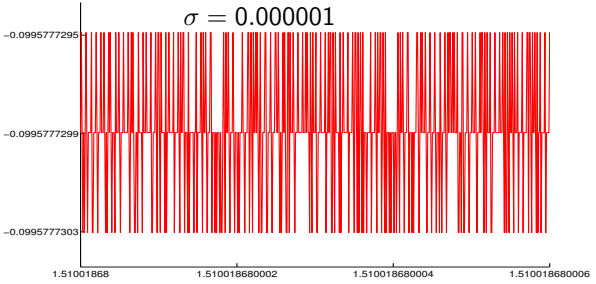
# Inaccurately evaluated integrand

Test case 1



# Inaccurately evaluated integrand

## Fast regime switching between `double` and `vpa`

**Algorithm 1** Fast regime switching algorithm for the evaluation of the integrand $f(k)$ that is by default in `double`.

$o_1 := \log_{10} |\Re(C_1)|$; $o_2 := \log_{10} |\Re(C_2)|$; $o := o_1 - o_2$; $\omega_0 = 22$;
$f_0 := f(0 + i/2)$;
`par` = **false**;
**if** $f_0 > 10^{-3}$ **and** $o > \omega_0$ **then**
  $\omega_1 := \min(5, \max(4 - \log_{10}(K/3), 0))$;
  $\omega_2 := \min(\log_{10} |\Re(f_0)|, 0)$;
  **if** $o > \omega_0 + \omega_1 - \omega_2$ **then**
    `par` = **true**;
  **end if**
**end if**
**if** `par` **then**
  switch the evaluation of the integrand $f(k)$ from `double` to `vpa`
**end if**

# Quadratures comparison

Test case 1: quadratures comparison

| method (case $\sigma = 0.001$) | value | error | time [s] | fevals |
|---|---|---|---|---|
| `integral` | 0.77681485 | 0.00000006 | 0.056 | 300 |
| `integral (vpa)` | 0.77681478 | 0.00000000 | 4.948 | 270 |
| `legendre (128)` | 0.77680533 | 0.00000944 | 0.006 | 64 |
| `legendre (256)` | 0.77682245 | 0.00000767 | 0.013 | 128 |
| `quad` | 0.77681499 | 0.00000020 | 0.092 | 1694 |
| `quadl` | 0.94027672 | 0.16346193 | 0.150 | 2018 |
| `quadl (vpa)` | 0.77681478 | 0.00000000 | 4.110 | 146 |
| `adaptlob` | 0.77681496 | 0.00000017 | 1.485 | 30734 |
| `trapz (0.01)` | 0.77681499 | 0.00000021 | 0.014 | 1 |
| `trapz (0.001)` | 0.77681499 | 0.00000020 | 0.027 | 1 |
| `trapz (vpa, 0.01)` | 0.77681478 | 0.00000000 | 14.287 | 1 |
| `trapz (vpa, 0.001)` | 0.77681478 | 0.00000000 | 301.293 | 1 |

# Quadratures comparison

Test case 1: quadratures comparison

| method (case $\sigma = 0.0001$) | value | error | time [s] | fevals |
|---|---|---|---|---|
| `integral` | 0.77683946 | 0.00002467 | 6.761 | 360780 |
| `integral (vpa)` | 0.77681478 | 0.00000000 | 4.872 | 270 |
| `legendre (128)` | 0.77695229 | 0.00013750 | 0.006 | 64 |
| `legendre (256)` | 0.77682884 | 0.00001405 | 0.012 | 128 |
| `quad` | 0.77688362 | 0.00006883 | 0.218 | 5006 |
| `quadl` | 0.75020018 | 0.02661460 | 0.146 | 2030 |
| `quadl (vpa)` | 0.77681478 | 0.00000000 | 4.101 | 146 |
| `adaptlob` | 0.77684007 | 0.00002528 | 2.278 | 47324 |
| `trapz (0.01)` | 0.77684249 | 0.00002770 | 0.010 | 1 |
| `trapz (0.001)` | 0.77683949 | 0.00002470 | 0.029 | 1 |
| `trapz (vpa, 0.01)` | 0.77681478 | 0.00000000 | 13.743 | 1 |
| `trapz (vpa, 0.001)` | 0.77681478 | 0.00000000 | 288.313 | 1 |

# Quadratures comparison

Test case 1: quadratures comparison

| method (case $\sigma = 0.00001$) | value | error | time [s] | fevals |
|---|---|---|---|---|
| `integral` | 0.76823548 | 0.00857930 | 0.655 | 28260 |
| `integral (vpa)` | 0.77681478 | 0.00000000 | 5.004 | 270 |
| `legendre (128)` | 0.76877555 | 0.00803923 | 0.007 | 64 |
| `legendre (256)` | 0.76810697 | 0.00870781 | 0.015 | 128 |
| `quad` | 0.76823622 | 0.00857856 | 0.099 | 2022 |
| `quadl` | 0.76846917 | 0.00834561 | 0.149 | 2018 |
| `quadl (vpa)` | 0.77681478 | 0.00000000 | 4.084 | 146 |
| `adaptlob` | 0.76823478 | 0.00858000 | 0.216 | 4232 |
| `trapz (0.01)` | 0.76828142 | 0.00853336 | 0.013 | 1 |
| `trapz (0.001)` | 0.76823697 | 0.00857781 | 0.030 | 1 |
| `trapz (vpa, 0.01)` | 0.77681478 | 0.00000000 | 14.618 | 1 |
| `trapz (vpa, 0.001)` | 0.77681478 | 0.00000000 | 310.098 | 1 |

# Quadratures comparison

Test case 1: switching regime `integral`

| $\sigma$ | $o$ | $\omega_1$ | par | `integral` / `integral (vpa)` | | | error |
|---|---|---|---|---|---|---|---|
| | $f_0$ | $\omega_2$ | | value | time | fevals | |
| 0.001 | 21.572 | – | false | 0.77681485 | 0.056 | 300 | 0.00000006 |
| | 2.137 | – | | 0.77681478 | 4.948 | 270 | |
| 0.0005 | 22.775 | 0.681 | true | 0.77681452 | 4.241 | 115320 | 0.00000025 |
| | 2.137 | 0 | | 0.77681478 | 2.244 | 270 | |
| 0.0001 | 25.568 | 0.681 | true | 0.77683946 | 6.761 | 360780 | 0.00002467 |
| | 2.1371 | 0 | | 0.77681478 | 4.872 | 270 | |
| 0.00005 | 26.715 | 0.681 | true | 0.77684142 | 4.198 | 217170 | 0.00002663 |
| | 2.1389 | 0 | | 0.77681478 | 5.056 | 270 | |
| 0.00001 | `Inf` | 0.681 | true | 0.76823548 | 0.655 | 28260 | 0.00857930 |
| | 2.1243 | 0 | | 0.77681478 | 5.004 | 270 | |
| 0.000001 | `Inf` | 0.681 | true | 0.77674994 | 0.068 | 750 | 0.00006484 |
| | 2.1243 | 0 | | 0.77681478 | 5.043 | 270 | |

# Quadratures comparison

## Test case 1: optimal switching regime

| method | | value | error | time [s] | fevals |
|---|---|---|---|---|---|
| ($\sigma = 0.0005$, $o = 22.775$, $f_0 = 2.137$, $\omega_1 = 0.681$, $\omega_2 = 0$, par = true) | | | | | |
| `integral` | | 0.77681452 | 0.00000025 | 2.244 | 115380 |
| `integral (vpa)` | | 0.77681478 | 0.00000000 | 4.920 | 270 |
| `integral (opt)` | ✓ | 0.77681478 | 0.00000000 | 4.157 | 270 |
| `quad (opt)` | ✓ | 0.77681478 | 0.00000000 | 6.206 | 304 |
| `quadl (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.520 | 146 |
| `adaptlob (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.668 | 164 |
| `vpaintegral` | | 0.77681305 | 0.00000172 | 7.694 | - |
| ($\sigma = 0.0001$, $o = 25.568$, $f_0 = 2.1371$, $\omega_1 = 0.681$, $\omega_2 = 0$, par = true) | | | | | |
| `integral` | | 0.77683946 | 0.00002467 | 6.761 | 360780 |
| `integral (vpa)` | | 0.77681478 | 0.00000000 | 4.872 | 270 |
| `integral (opt)` | ✓ | 0.77681478 | 0.00000000 | 4.112 | 270 |
| `quad (opt)` | ✓ | 0.77681478 | 0.00000000 | 6.270 | 304 |
| `quadl (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.520 | 146 |
| `adaptlob (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.467 | 164 |
| `vpaintegral` | | 0.77685918 | 0.00004440 | 31.627 | - |

# Quadratures comparison

## Test case 1: optimal switching regime

| method | | value | error | time [s] | fevals |
|---|---|---|---|---|---|
| ($\sigma = 0.00005$, $o = 26.715$, $f_0 = 2.1389$, $\omega_1 = 0.681$, $\omega_2 = 0$, par = true) | | | | | |
| `integral` | | 0.77684142 | 0.00002663 | 4.198 | 217170 |
| `integral (vpa)` | | 0.77681478 | 0.00000000 | 5.056 | 270 |
| `integral (opt)` | ✓ | 0.77681478 | 0.00000000 | 4.165 | 270 |
| `quad (opt)` | ✓ | 0.77681478 | 0.00000000 | 6.317 | 304 |
| `quadl (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.615 | 146 |
| `adaptlob (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.480 | 164 |
| `vpaintegral` | | 0.77682241 | 0.00000762 | 31.843 | - |
| ($\sigma = 0.00001$, $o =$ `Inf`, $f_0 = 2.1243$, $\omega_1 = 0.681$, $\omega_2 = 0$, par = true) | | | | | |
| `integral` | | 0.76823548 | 0.00857930 | 0.655 | 28260 |
| `integral (vpa)` | | 0.77681478 | 0.00000000 | 5.004 | 270 |
| `integral (opt)` | ✓ | 0.77681478 | 0.00000000 | 4.108 | 270 |
| `quad (opt)` | ✓ | 0.77681478 | 0.00000000 | 6.391 | 304 |
| `quadl (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.594 | 146 |
| `adaptlob (opt)` | ✓ | 0.77681478 | 0.00000000 | 3.582 | 164 |
| `vpaintegral` | | 0.76839158 | 0.0084231 | 13.270 | - |

# Conclusion and further issues

Numerical analysis:

- evaluation of the integrand in `vpa` is time consuming,
- we designed a suitable fast algorithm that could tell if the integrand is `double` sufficient or if it has to be evaluated in higher precision,
- we compared numerical quadratures especially for problematic (`double` insufficient) cases that has to be switched to `vpa`,
- idea of `vpaintegral` is promissing, it can precisely calculate simple integrands, but still has serious problems with the studied integrand.

Further issues:

- high precision in other financial applications.

Thank you for your attention!