CT/RJ-Mix Transdimensional MCMC for Gaussian mixtures

Olivier Cappé, Tobias Rydén, Christian P. Robert

Version 3.6, October 6, 2005

Contents

1 Description and version information

This is the C code that was used to produce the figures in Section 4 of

> O. Cappé, C. Robert, and T. Rydén. *Reversible jump, birth-and-death and more general continuous time Markov chain Monte Carlo samplers.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), Volume 65, Issue 3, pages 679-700, 2003.

The purpose of this software is to implement transdimensional Markov Chain Monte Carlo (MCMC) for inference in (scalar) Gaussian mixture models, with unknown number of components. Two methods are implemented: Reversible Jump (RJ) MCMC for `rj_mix` and Continuous Time (CT) for `ct_mix`. See the above mentioned article for further details and references about the method.

The current version of the software is 3.6 as printed on all pages of this manual. This is the second version that is made publicly available. Changes from the first version include removal of misleading comments in `alea.c` and correction of an error in `Gam` which made sampling from gamma distribution poorly reliable when a high value of the parameter was used. Each file has its own individual version number (which differs from 3.6).

2 How to

2.1 Obtain the source code

Download it from http://www.tsi.enst.fr/˜cappe/ctrj_mix/

2.2 Build the software

After unpacking the archive, enter the main directory (named `ctrj_mix`) and type `make`, you should obtain the two executables `rj_mix` and `ct_mix`.

2.3 Check that it works

Cd to the sub-directory `demo` and type `make demo`. This will run the two programs `rj_mix` and `ct_mix` on the data in `galaxy.dat` (this is the same centered galaxy dataset that was used in the paper), using the settings found in `galaxy.arg`. You should see on the screen something like this

```
$ make demo
../rj_mix galaxy
Data has 82 samples. Running 5000 x 8 iterations of the sampler...
SK0=1179069493
SK1=2853338688
SK2=1059189655
[snip]
../ct_mix galaxy
Data has 82 samples. Running 5000 x 8 iterations of the sampler...
SK0=469243900
SK1=500366318
SK2=168522948
[snip]
```

The following files will be created in the `demo` sub-directory:

```
galaxy.sts
galaxy.res
galaxy.st2
galaxy.rs2
```

`galaxy.res` and `galaxy.rs2` are binary files whose coding may vary depending on the type of machine (big-endian or low-endian) that you are using. On the other hand, `galaxy.sts` and galaxy.st2 are text files that contain summary information on the output of the two program, the content of these should not depend on your system, which you can check by running

```
$ make check
md5sum galaxy.sts
10b31093f6bf13acc531dcbe04d22d42  galaxy.sts
md5sum galaxy.st2
d55c5667ebf0913263c8eca8b68c7b13  galaxy.st2
```

(note that this will only work if you have the md5sum command installed on your system). Another easy test is that the SK0=..., SK1=... and SK2=... lines in the text output of the programs (see above) are the values of the seeds of the random generator after the program run. These should also correspond to the excerpt given above.

On a 2 GHz Pentium IV machine the make demo command takes about 50s to complete (each program performs 40 000 iterations, ct_mix being much slower than rj_mix for reasons discussed in the paper).

2.4 Troubleshooting

The validity of the previous comments have been checked on several Intel-x86 based PCs running various versions of GNU/Linux or using Cygwin within Microsoft Windows. It has also been tested on SPARC workstations running Sun Microsystems SunOS 5.8. It may be the case however that for some other architectures, different implementations of the basic C functions could render the output somewhat different. A possible source of problem would be if unsigned long int corresponded to something else than 4 bytes variables (see the comments at the beginning of the alea.h file).

If you don't pass the tests above, the next thing to try is to check the output to see if you obtain a figure comparable to the one shown in Section 2.5 below.

2.5 Understand the program output

The screen output by rj_mix and ct_mix is a copy of the content of the input .arg file (galaxy.arg in the case of the demo) except for the fact that the SK0=..., SK1=... and SK2=... lines are updated to the values of the seeds of the random generator after the program run. This can be useful if you intend to run the program several times independently and want to be able to reproduce the whole experiment.

The output formats of the two programs differ slightly:

rj_mix The statistics file (.sts) is a text file with one line for each iteration:

```
| n iter | k | log-l | move type |
                                 |
       for type 1 (fixed k):     | accpt w | accpt mu | accpt var  |
       for type 2 (birth/death): | birth   | accpt    | unused (-1)|
       for type 3 (split/merge): | split   | accpt    | unused (-1)|
```

The results file (.res) is a binary file which contains floating point numbers (in C double format). The results file is a succession of records, one for each recorded iteration, starting with the initial value, where the records have the following structure:

```
w(i,1), ..., w(i,k(i)), mu(i,1), ..., mu(i,k(i)), var(i,1), ... var(i,k(i))
```

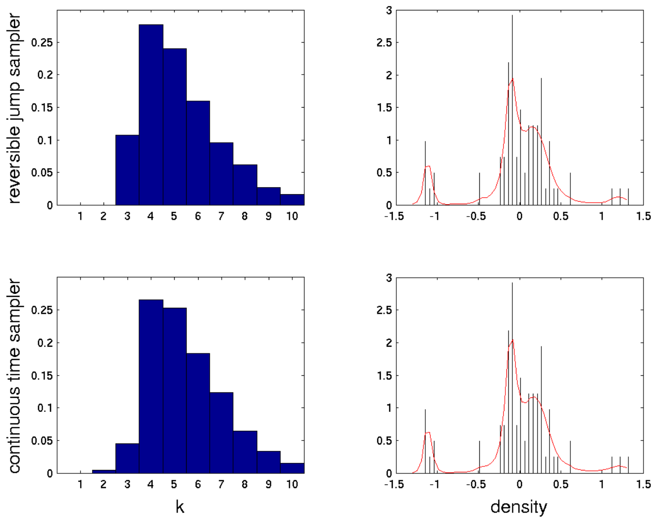
where `i` is the number of the iteration (`i=0` is the initialization, and for `i>0`, `i*SubSamp` is the corresponding iteration index). Each record is thus of length `3*k(i)`, which varies with `k`. In order to read the `.res` file, it is thus necessary to use the second column of the corresponding `.sts` file which gives the number of components for each iteration.

ct_mix The statistics file (`.st2`) is a text file with one line for each iteration:

```
| n iter | k | log-l | move type |
                                 |
       for type 1 (fixed k):     | base2(accpt w,mu,var) | weight |
       for type 2 (birth/death): | birth                 | weight |
       for type 3 (split/merge): | split                 | weight |
```

The structure of the results file (`.rs2`) is similar to that of `rj_mix` with varying-length records, except that each record has a leading value which corresponds to the weight (inverse duration) of the simulated state.

If you work with OCTAVE or MATLAB, you can use the provided functions `readres.m` and `readres2.m` to load the outputs of, respectively `rj_mix` and `ct_mix`. The script `galaxy.m` can be used to display some properties of the output obtained when running `make demo`. It should display the following plot (in MATLAB only)



2.6 Specify you own input

When typing `make demo`, the two programs use the arguments contained in the text file `galaxy.arg` (where each program is called with the name `galaxy`, without extension, as argument). The structure of this file should be easy to understand:

SK0, SK1, SK2 Three seeds of the random generator (3 integer values)

NOut Number of iterations recorded in the result file (the initialization state will be added).

SubSamp Subsampling factor, the total number of iterations is `NOut*SubSamp`.

Data Name of the file that contains the data (should be a plain text file, with one number per line).

M Maximum number of components.

Kappa Hyperparameter for the means (floating point value), see κ below.

AlphaVar Hyperparameter for the variances (floating point value), see α below.

BetaVar Hyperparameter for the variances (floating point value), see β below.

Eta Parameter of the fixed-k move for updating the weights (floating point value), see η below.

Rho Parameter of the fixed-k move for updating the means (floating point value), see ρ below.

Nu Parameter of the fixed-k move for updating the variances (floating point value), see ν below.

PFixed Probability of attempting the fixed-k moves (the third types of fixed-k moves are tried in turn).

PBirth Probability of attempting a birth move.

PDeath Probability of attempting a death move.

PSplit Probability of attempting a split move.

Gamma_S Parameter of the split proposal for the weights (floating point value), see γ_S below.

Rho_S Parameter of the split proposal for the means (floating point value), see ρ_S below.

Nu_S Parameter of the split proposal for the variances (floating point value), see ν_S below.

The probabilities should add to less than one since `1-PFixed-PBirth-PSplit` will be the probability of attempting a merge move. They can be set to zero, meaning that the corresponding move will never be attempted.

`ct_mix` use `PFixed`, `PBirth` and `PSplit` as rates for the corresponding type of moves and hence only their relative values are of some importance (`ct_mix` discards `PDeath` as the death and split rates have to be updated, based on the simulated configuration, when doing continuous time sampling).

Be aware that the function that reads the `.arg` file is pretty primitive and won't tolerate neither spaces before or after the `=` sign nor case variations. The scanning of the file ends after the `Nu_S=...` line and you can thus write whatever you want after this line.

2.7 Known limitations

The programs are reasonably optimized but certainly not overly sophisticated. In particular, all arrays are allocated statically, with their size fixed at compile time. The program will thus only accept data files with less than 2000 observations and the maximum value of `M` (maximum number of components) is set to 20. This can be changed by editing the top section of `mix_lib.h`. If you do so, please take so time to read the comments and modify the value of `MaxKPairs` accordingly. Note that if you really have a lot of components, you should avoid using `ct_mix` for reasons discussed in the paper.

3 Technical appendix

3.1 Model

3.1.1 Likelihood

Gaussian scalar mixture model with parameters $(w_{1:k}, \mu_{1:k}, \upsilon_{1:k})$ **(by convention** υ_i **are the variances and** Normal$(0, \rho)$ **has <u>variance</u>** ρ**)**

$$\log l(Y_{1:n}|\boldsymbol{\theta}) = \sum_{t=1}^{n} \log \left(\sum_{i=1}^{k} \frac{w_i}{\sqrt{2\pi\upsilon_i}} \exp\left[-\frac{(Y_t - \mu_i)^2}{2\upsilon_i} \right] \right) \tag{1}$$

3.1.2 Prior

- $k \sim$ Uniform$\{1, \dots M\}$
- $w_{1:k} \sim$ Dirichlet$(1, \dots, 1)$ with pdf $(k-1)!$ on the simplex
- $\mu_{1:k} \sim \otimes$ Normal$(0, \kappa)$ with log-pdf

$$\log p(\mu_{1:k}|\kappa) = -\frac{k}{2}\log(2\pi\kappa) - \frac{1}{2\kappa}\sum_{i=1}^{k}\mu_i^2 \tag{2}$$

- $\upsilon_{1:k} \sim \otimes$ Inverse-Gamma(α, β) with log-pdf

$$\log p(\upsilon_{1:k}|\kappa) = k(\alpha\log\beta - \log\Gamma(\alpha)) - \sum_{i=1}^{k}\left[(\alpha+1)\log(\upsilon_i) + \frac{\beta}{\upsilon_i}\right] \tag{3}$$

3.1.3 Values of the hyperparameters

For the `galaxy.dat` dataset,

$$M = 15 \tag{4}$$
$$\kappa = (\max\{Y_i\}_{1\le i\le n} - \min\{Y_i\}_{1\le i\le n})^2 \tag{5}$$
$$\alpha = 0.5 \tag{6}$$
$$\beta = 10^{-3} \tag{7}$$

are reasonably "non-informative" values, without being absolutely dispersed (note: birth moves are simulated from the prior).

3.2 Fixed k moves

3.2.1 The moves

They are proposed with probability $P_F(k)$. We don't use completion (ie. simulating the indicator variables) on these moves, which are random walk like and proposed each in turn:

For $w_{1:k}$ ▶ Normalized multiplicative RW MH with multiplicative distribution $\otimes$ Log-Normal$(0, \eta)$. The acceptance rate is (see Section 3.5 below)

$$\log l(Y_{1:n}|\tilde{\boldsymbol{\theta}}) - \log l(Y_{1:n}|\boldsymbol{\theta}) + \sum_{i=1}^{n}\log\frac{\tilde{w_i}}{w_i} \tag{8}$$

For $\mu_{1:k}$ ▶ RW MH with perturbation distribution $\otimes$ Normal$(0,\rho)$.
The acceptance ratio is just

$$\log l(Y_{1:n}|\tilde{\boldsymbol{\theta}}) - \log l(Y_{1:n}|\boldsymbol{\theta}) - \sum_{i=1}^{k} \frac{\tilde{\mu}_i^2 - \mu_i^2}{2\kappa} \wedge 0 \tag{9}$$

and the move can just as easily be carried out globally or one component at a time.

For $\upsilon_{1:k}$ ▶ Multiplicative RW MH with multiplicative distribution $\otimes$ Log-Normal$(0,\nu)$.
The acceptance ratio is

$$\frac{l(Y_{1:n}|\tilde{\boldsymbol{\theta}})}{l(Y_{1:n}|\boldsymbol{\theta})} \times \frac{\pi(\tilde{\boldsymbol{\theta}})}{\pi(\boldsymbol{\theta})} \times \prod_{i=1}^{n} \frac{\frac{1}{\upsilon_i}\text{Normal}(\log \upsilon_i | \log \tilde{\upsilon}_i, \nu)}{\frac{1}{\tilde{\upsilon}_i}\text{Normal}(\log \tilde{\upsilon}_i | \log \upsilon_i, \nu)} \wedge 1 \tag{10}$$

or in log

$$\log l(Y_{1:n}|\tilde{\boldsymbol{\theta}}) - \log l(Y_{1:n}|\boldsymbol{\theta}) + \sum_{i=1}^{n} \left(-(\alpha+1)\log\frac{\tilde{\upsilon}_i}{\upsilon_i} - \beta(\frac{1}{\tilde{\upsilon}_i} - \frac{1}{\upsilon_i}) \right) + \sum_{i=1}^{n} \log(\frac{\tilde{\upsilon}_i}{\upsilon_i}) \wedge 0 \tag{11}$$

The move can just as easily be carried out globally or one component at a time.

For the continuous time sampler, the fixed k move is proposed with rate $\lambda_F(k) \propto P_F(k)$. As in the RJMCMC case, it consists of the three MH proposals (weights, means, variances) with independent accept/reject decisions.

3.2.2 Tuning the sampler parameters

Experimenting with the moves above on the `galaxy.dat` data set, it is found that the global version of the moves with

$$\begin{cases} \eta &= 0.05 \\ \rho &= \dfrac{\kappa}{2000 \times k} \\ \nu &= 0.08 \end{cases} \tag{12}$$

is satisfying (with acceptance rates that stay in the range 0.3-0.7 for all values of $k \leq 15$). The normalization of ρ by k tends to stabilize the acceptance rate (with constant ρ, the acceptance drops for high values of k).

3.3 Birth or death moves

When in a k components configuration, we propose a new component from the prior according to

1. $\tilde{w}_{k+1} \sim$ Beta$(1,k)$ with $\tilde{w}_{1:k+1} = ((1-\tilde{w}_{k+1})w_{1:k}, \tilde{w}_{k+1})$
2. $\tilde{\mu}_{k+1} \sim$ Normal$(0,\kappa)$
3. $\tilde{\upsilon}_{k+1} \sim$ Inverse-Gamma(α,β)

The acceptance ratio for the corresponding RJ move is

$$\frac{l(Y_{1:n}|\tilde{\boldsymbol{\theta}}_{k+1})}{l(Y_{1:n}|\boldsymbol{\theta}_k)} \times \frac{P_D(k+1)}{P_B(k)} \wedge 1 \tag{13}$$

For the continuous time version of the move, if the birth rate is $\lambda_B(k) \propto P_B(k)$, the death rates are given by

$$\frac{l(Y_{1:n}|\boldsymbol{\theta}_k)}{l(Y_{1:n}|\tilde{\boldsymbol{\theta}}_{k+1})} \times \frac{\lambda_B(k)}{k+1} \tag{14}$$

This is basically Stephens (2000) proposal.

In the current implementation, $\lambda_F = P_F$, $\lambda_B = P_B$ and $P_D = (1-(P_F+P_B))$ which are all constant (ie. do not vary with k), except for obvious edge effects ($P_D(1) = 0$, $P_B(M) = 0$).

3.4 Split or merge moves

3.4.1 Split proposal

When in a k components configuration, we propose to split component i according to

1. $w_i \longrightarrow (\tilde{w}_i, \tilde{\tilde{w}}_i) = (\xi w_i, (1-\xi)w_i)$ with $\xi \sim \text{Beta}(\gamma_S, \gamma_S)$ (with $\gamma_S \approx 2$)
2. $\mu_i \longrightarrow (\tilde{\mu}_i, \tilde{\tilde{\mu}}_i) = (\mu_i - \xi, \mu_i + \xi)$ with $\xi \sim \text{Normal}(0, \rho_S)$
3. $\upsilon_i \longrightarrow (\tilde{\upsilon}_i, \tilde{\tilde{\upsilon}}_i) = (\upsilon_i/\xi, \upsilon_i\xi)$ with $\xi \sim \text{Log-Normal}(0, \nu_S)$

3.4.2 Acceptance ratio

The acceptance ratio for a split move in $\min(1, A)$ where

$$\begin{aligned} A &= \frac{l(Y_{1:n}|\tilde{\boldsymbol{\theta}}_{k+1})}{l(Y_{1:n}|\boldsymbol{\theta}_k)} \times \frac{(k+1)!\, p(\tilde{\boldsymbol{\theta}}_{k+1})}{k!\, p(\boldsymbol{\theta}_k)} \\ &\quad \times \frac{P_M(k+1)/(k(k+1)/2)}{P_S(k)/k} \times (2 \text{ proposal pdf})^{-1} \times \left|\frac{\partial\, \text{new}}{\partial\, \text{old}}\right| \\ &= \frac{l(Y_{1:n}|\tilde{\boldsymbol{\theta}}_{k+1})}{l(Y_{1:n}|\boldsymbol{\theta}_k)} \times \frac{P_M(k+1)}{P_S(k)} \times \underbrace{\frac{p(\tilde{\boldsymbol{\theta}}_{k+1})}{p(\boldsymbol{\theta}_k)}}_{T_1} \times \underbrace{\text{proposal pdf}^{-1}}_{T_2} \times \underbrace{\left|\frac{\partial\, \text{new}}{\partial\, \text{old}}\right|}_{T_3} \end{aligned}$$

with

Weights

$$T_1 = k \tag{15}$$

$$T_2 = \left(\frac{\Gamma(2\gamma_S)}{\Gamma(\gamma_S)^2} \frac{\tilde{w}_i^{\gamma_S-1}\tilde{\tilde{w}}_i^{\gamma_S-1}}{w_i^{2(\gamma_S-1)}}\right)^{-1} \tag{16}$$

$$T_3 = w_i \tag{17}$$

Means

$$T_1 = \frac{1}{\sqrt{2\pi\kappa}} \exp\left[\frac{\mu_i^2 - (\tilde{\mu}_i^2 + \tilde{\tilde{\mu}}_i^2)}{2\kappa}\right] \tag{18}$$

$$T_2 = \sqrt{2\pi\rho_S} \exp\left[\frac{(\tilde{\tilde{\mu}}_i - \tilde{\mu}_i)^2}{8\rho_S}\right] \tag{19}$$

$$T_3 = 2 \tag{20}$$

Variances

$$T_1 = \frac{\beta^\alpha}{\Gamma(\alpha)} \upsilon_i^{-(\alpha+1)} \exp\left[\beta\left(\upsilon_i^{-1} - \tilde{\upsilon}_i^{-1} - \tilde{\tilde{\upsilon}}_i^{-1}\right)\right] \tag{21}$$

$$T_2 = \sqrt{2\pi\nu_S}\sqrt{\tilde{\tilde{\upsilon}}_i/\tilde{\upsilon}_i} \exp\left[\left(\log\frac{\tilde{\tilde{\upsilon}}_i}{\tilde{\upsilon}_i}\right)^2/(8\nu_S)\right] \tag{22}$$

$$T_3 = 2\frac{\upsilon_i}{\sqrt{\tilde{\tilde{\upsilon}}_i/\tilde{\upsilon}_i}} \tag{23}$$

3.4.3 Merge rate for the continuous time version

If $\lambda_S(k)$ denotes the split rate, the merge rate is given by

$$\frac{l(Y_{1:n}|\tilde{\boldsymbol{\theta}}_k)}{l(Y_{1:n}|\boldsymbol{\theta}_{k+1})} \times \frac{2\lambda_S(k)}{k(k+1)} \times (T_1 \times T_2 \times T_3)^{-1}$$

where T_1, T_2 and T_3 are as given above.

3.4.4 Choice of the sampler parameters

In the current implementation, $P_S(k)$ and $P_M(k)$ are constant (with k), except for edge effects ($P_M(1) = 0$, $P_S(M) = 0$). On the `galaxy.dat` dataset, the choice of parameters that maximizes the rate of acceptance of the split or merge move is

$$\gamma_S = 1 \tag{24}$$

$$\rho_S = 0.2 \tag{25}$$

$$\nu_S = 3 \tag{26}$$

But the acceptance probability is then only of 5% (compared to 15% for the simpler birth or death move).

3.5 Appendix: Formulas for normalized log-normal multiplicative RW

Consider the following proposal:

1. Generate $\xi_1, \ldots, \xi_k$ from $\otimes\,\text{Log-Normal}(\log w_i, \tau^2)$ distribution
2. Take

$$\begin{aligned} \varpi &= \xi_1 + \ldots + \xi_k \\ \tilde{w}_1 &= \xi_1/\varpi \\ &\ldots \\ \tilde{w}_k &= \xi_k/\varpi \end{aligned}$$

The proposal density can be obtained by the change of variable formula. Indeed, $(\varpi, \tilde{w}_1, \ldots, \tilde{w}_{k-1})$ is then distributed from the density

$$\frac{1}{(2\pi\tau^2)^{k/2}} \prod_{i=1}^{k} \frac{\exp\{-(\log w_i - \log[\varpi\tilde{w}_i])^2/2\tau^2\}}{\varpi\tilde{w}_i} \times \left|\frac{\partial(\xi_1, \ldots, \xi_k)}{\partial(\varpi, \tilde{w}_1, \ldots, \tilde{w}_{k-1})}\right|$$

and

$$\begin{aligned}
\left|\frac{\partial(\xi_1,\dots,\xi_k)}{\partial(\varpi,\tilde w_1,\dots,\tilde w_{k-1})}\right| &= \begin{vmatrix} \tilde w_1 & \tilde w_2 & \dots & \tilde w_k \\ \varpi & 0 & \dots & -\varpi \\ 0 & \varpi & \dots & -\varpi \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -\varpi \end{vmatrix} \\
&= \varpi^{k-1} \begin{vmatrix} \tilde w_1 & \tilde w_2 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{vmatrix} \\
&= \varpi^{k-1}\,.
\end{aligned}$$

Thus,

$$(\varpi,\tilde w_1,\dots,\tilde w_{k-1}) \sim \frac{1}{(2\pi\tau^2)^{k/2}\varpi}\prod_{i=1}^k \frac{\exp\{-(\log w_i - \log\varpi - \log\tilde w_i])^2/2\tau^2\}}{\tilde w_i}$$

Now, if we denote $\zeta = \log(\varpi)$, ζ is distributed as

$$\prod_{i=1}^k \exp\left\{-(\log w_i - \log\tilde w_i - \zeta)^2/2\tau^2\right\}$$

(up to a multiplicative constant) conditionally on the $\tilde w_i$'s. This is a normal density, meaning that

$$\zeta|\tilde w_1,\dots,\tilde w_k \sim \mathcal{N}\left(\frac{1}{k}\sum_{i=1}^k(\log w_i - \log\tilde w_i), \frac{\tau^2}{k}\right)$$

Therefore the marginal distribution of the vector $(\tilde w_1,\dots,\tilde w_{k-1})$ is available in closed form, that is

$$\begin{aligned}(\tilde w_1,\dots,\tilde w_{k-1}) \sim \prod_{i=1}^k & \frac{\exp\left\{-(\log w_i - \log\tilde w_i)^2/2\tau^2\right\}}{\tilde w_i} \\ & \times \exp\left\{\left(\sum_{i=1}^k(\log w_i - \log\tilde w_i)\right)^2 \Big/ 2k\tau^2\right\} \end{aligned} \tag{27}$$

up to a multiplicative constant which is equal to

$$\frac{1}{(2\pi\tau^2)^{k/2}} \times \frac{\sqrt{2\pi}\tau}{\sqrt{k}} = \frac{1}{(2\pi\tau^2)^{(k-1)/2}\sqrt{k}}\,.$$