RWTH Aachen

Department of Computer Science
Technical Report

Symbolic vs. Algorithmic Differentiation of GSL Integration Routines

Niloofar Safiran, Uwe Naumann

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2015-14

RWTH Aachen · Department of Computer Science · November 2015

Symbolic vs. Algorithmic Differentiation of GSL Integration Routines

Niloofar Safiran, Uwe Naumann

LuFG Informatik 12: Software and Tools for Computational Engineering,
RWTH Aachen University, Germany.
Email: {safiran, naumann}@stce.rwth-aachen.de

Abstract. Forward and reverse modes of algorithmic differentiation (AD) transform implementations of multivariate vector functions $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ as computer programs into tangent and adjoint code, respectively. The adjoint mode is of particular interest in large-scale functions due to the independence of its computational cost on the number of free variables. The additional memory requirement for the computation of derivatives of the output with respect to parameters by a fully algorithmic method (derived by AD) can quickly become prohibitive for large values of n. This can be reduced significantly by the symbolic approach to differentiation of the underlying integration routine. Vectorizing gsl routines for integration and applying symbolic adjoint on them has considerably less memory requirement with nearly the same runtime overhead and in most cases faster convergence in comparison with algorithmic adjoint.

1 Differentiation of Integrals

Let us consider an interval which the limits of the integral are themselves functions of $\alpha \in \mathbb{R}$, it follows that:

$$I(\alpha) = \int_{a(\alpha)}^{b(\alpha)} f(\alpha, x)dx = F(\alpha, b(\alpha)) - F(\alpha, a(\alpha)) \quad , \tag{1}$$

which yields the partial derivatives

$$\frac{\partial I}{\partial b} = f(\alpha, b(\alpha)) \quad , \quad \frac{\partial I}{\partial a} = -f(\alpha, a(\alpha)) \quad .$$

Considering chain rule and Leibniz's rule for differentiation under the integral sign [Fla73],

$$\frac{dI}{d\alpha} = f(\alpha, b(\alpha))\frac{db}{d\alpha} - f(\alpha, a(\alpha))\frac{da}{d\alpha} + \int_{a(\alpha)}^{b(\alpha)} \frac{\partial f(\alpha, x)}{\partial \alpha}dx \quad . \tag{2}$$

Now suppose that, $\alpha \in \mathbb{R}^n$, i.e.,

$$I(\alpha_1, \alpha_2, \ldots, \alpha_n) = \int_{a(\alpha_1,\alpha_2,\ldots,\alpha_n)}^{b(\alpha_1,\alpha_2,\ldots,\alpha_n)} f(\alpha_1, \alpha_2, \ldots, \alpha_n, x)dx \quad . \tag{3}$$

Differentiating the above equation with respect to all parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ yields:

$$
\begin{aligned}
\frac{dI}{d\alpha_1} =& f(\boldsymbol{\alpha}, b(\boldsymbol{\alpha}))\frac{db(\boldsymbol{\alpha})}{d\alpha_1} - f(\boldsymbol{\alpha}, a(\boldsymbol{\alpha}))\frac{da(\boldsymbol{\alpha})}{d\alpha_1} + \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \frac{\partial f(\boldsymbol{\alpha}, x)}{\partial \alpha_1} dx \quad , \\
\frac{dI}{d\alpha_2} =& f(\boldsymbol{\alpha}, b(\boldsymbol{\alpha}))\frac{db(\boldsymbol{\alpha})}{d\alpha_2} - f(\boldsymbol{\alpha}, a(\boldsymbol{\alpha}))\frac{da(\boldsymbol{\alpha})}{d\alpha_2} + \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \frac{\partial f(\boldsymbol{\alpha}, x)}{\partial \alpha_2} dx \quad , \\
& \dots \\
\frac{dI}{d\alpha_n} =& f(\boldsymbol{\alpha}, b(\boldsymbol{\alpha}))\frac{db(\boldsymbol{\alpha})}{d\alpha_n} - f(\boldsymbol{\alpha}, a(\boldsymbol{\alpha}))\frac{da(\boldsymbol{\alpha})}{d\alpha_n} + \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \frac{\partial f(\boldsymbol{\alpha}, x)}{\partial \alpha_n} dx \quad .
\end{aligned}
$$

In other words

$$
\nabla I = f(\boldsymbol{\alpha}, b(\boldsymbol{\alpha}))\nabla b - f(\boldsymbol{\alpha}, a(\boldsymbol{\alpha}))\nabla a + \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \nabla f(\boldsymbol{\alpha}, x) dx \quad . \tag{4}
$$

This means that we have gradients of bounds multiplied by integrand and one quadrature instead of gradient of quadrature.

2 Numerical Integration in GSL

In gsl [GDT+09], there are routines for adaptive and non-adaptive integration of general functions, with specialised routines for specific cases. These include integration over infinite and semi-infinite ranges, singular integrals, including logarithmic singularities, computation of Cauchy principal values and oscillatory integrals.

Each algorithm computes an approximation to a definite integral of the form,

$$
I = \int_a^b f(x)w(x)dx \quad ,
$$

where $w(x)$ is a weight function (for general integrands $w(x) = 1$). The user provides absolute and relative error bounds ($epsabs, epsrel$) which specify the following accuracy requirement,

$$
|RESULT - I| <= max(epsabs, epsrel|I|) \quad ,
$$

where $RESULT$ is the numerical approximation computed by the algorithm. The algorithms attempt to estimate the absolute error $ABSERR = |RESULT - I|$ in such a way that the following inequality holds,

$$
|RESULT - I| <= ABSERR <= max(epsabs, epsrel|I|) \quad .
$$

In short, the routines return the first approximation which has an absolute error smaller than $epsabs$ or a relative error smaller than $epsrel$.

The algorithms in QUADPACK use a naming convention based on the following letters,

– Q - quadrature routine

– N - non-adaptive integrator
– A - adaptive integrator
– G - general integrand (user-defined)
– W - weight function with integrand
– S - singularities can be more readily integrated
– P - points of special difficulty can be supplied
– I - infinite range of integration
– O - oscillatory weight function, cos or sin
– F - Fourier integral
– C - Cauchy principal value

The algorithms are built on combination of quadrature rules, a lower order rule and a higher order rule. The higher order rule is used to compute the best approximation of the integral over a small range. The difference between the results of the higher order rule and the lower order rule gives an estimate of the error in the approximation.

The *gsl_function* contains the value x as well as the parameters and is defined as

Listing 1.1: Definition of gsl_function with Parameters

```
struct gsl_function_struct{
  double (* function) (double x, void * params);
  void * params;
};
typedef struct gsl_function_struct gsl_function ;
#define GSL_FN_EVAL(F,x) (*((F)->function))(x,(F)->params)
```

The integration region in the adaptive integration algorithms in gsl is divided into subintervals, and on each iteration the subinterval with the largest estimated error is bisected. This reduces the overall error rapidly, as the subintervals become concentrated around local difficulties in the integrand. These subintervals are managed by a *gsl_integration_workspace* struct, which handles the memory for the subinterval ranges, results and error estimates.

Function: *gsl_integration_workspace * gsl_integration_workspace_alloc (size_t n_max)*

This function allocates a workspace sufficient to hold *n_max* double precision intervals, their integration results and error estimates.

Listing 1.2: workspace

```
typedef struct{
   size_t limit;
   size_t size;
   size_t nrmax;
   size_t i;
   size_t maximum_level;
  double *alist;
```

```
  double *blist;
  double *rlist;
  double *elist;
  size_t *order;
  size_t *level;}
gsl_integration_workspace;
```

In *gsl_workspace_alloc* function, *n_max* is the amount of memory allocated to workspace members *alist, blist, rlist, elist, order* and *level.*

2.1 Integrands Without Weight Functions

The algorithms for general functions (without a weight function) are based on Gauss-Kronrod rules. A Gauss-Kronrod rule begins with a classical Gaussian quadrature rule of order m. This is extended with additional points between each of the abscissae to give a higher order Kronrod rule of order $2m + 1$. The Kronrod rule is efficient because it reuses existing function evaluations from the Gaussian rule. The higher order Kronrod rule is used as the best approximation to the integral, and the difference between the two rules is used as an estimate of the error in the approximation.

2.2 Integrands With Weight Functions

For integrands with weight functions the algorithms use Clenshaw-Curtis quadrature rules. A Clenshaw-Curtis rule begins with an m-th order Chebyshev polynomial approximation to the integrand. This polynomial can be integrated exactly to give an approximation to the integral of the original function. The Chebyshev expansion can be extended to higher orders to improve the approximation and provide an estimate of the error.

2.3 Integrands With singular Weight Functions

The presence of singularities (or other behaviour) in the integrand can cause slow convergence in the Chebyshev approximation. The modified Clenshaw-Curtis rules used in QUADPACK separate out several common weight functions which cause slow convergence. These weight functions are integrated analytically against the Chebyshev polynomials to pre-compute modified Chebyshev moments. Combining the moments with the Chebyshev approximation to the function gives the desired integral. The use of analytic integration for the singular part of the function allows exact cancellations and substantially improves the overall convergence behaviour of the integration.

3 Algorithmic Differentiation of GSL Integration Routines

Algorithmic tangent and adjoint versions of the integration routine in gsl compute the directional derivatives of the approximation of the solution, which is actually computed by the algorithm [GW08,Nau12], in which AD is applied to the individual statements of the given implementation. In tangent mode, this yields an increase of roughly two in memory requirement as well as operation count. In the adjoint mode, data required within the reverse section is recorded in the forward section. The resulting memory requirement is likely to exceed the available resources for most real-world applications [1]. In the adjoint version, the number of operations is two times the operations (OPS) performed by the algorithm itself. The required memory in this case is proportional to the number of operations.

In order to apply AD tool [2] to the gsl integration routines, a separate library *integration-multidim* is built according to *integration* library in gsl but with the following changes:

- Include *dco.hpp* in gsl_math.h file, so that gsl knows the dco data types.
- Define the *gsl_function* with dco types $dco :: gt1s < double >:: type$ and $dco :: ga1s < double >:: type$ for tangent and adjoint version respectively.
- Define the related functions and routines with dco types.
- In some cases only the real value of the input is needed. In this case use the *get* function of dco. This returns the real part (double) of the input.

Note that gsl is written in C and dco is written in `C++`. In order to run dco in gsl, configure gsl with `g++`. For implementation set the right seed in the main function, call the integration routine and get the result of integration as well as the derivatives of the integral with respect to its parameters.

Listing 1.3: Algorithmic Tangent

```
gsl_integration_workspace_t1s_type* w =
        gsl_integration_workspace_alloc_t1s_type(100000);
struct my_f_params<dco::gt1s<double>::type> params;
for(int i=0; i<n; i++) {
    params.alpha = vec_alpha;
    dco::gt1s<double>::set(params.alpha[i], 1., 1);
    initialise_boundaries(a, b, params.alpha);
    gsl_function_t1s_type F;
    F.function = &func;
    F.params = &params;
    gsl_integration_qags(&F, a, b, 1e-7, 1e-7, w->limit,
                           w, &result, &error);
    dco::gt1s<double>::get(result, presult);
    dco::gt1s<double>::get(result, dresult, 1);
    printf("dI/da[%d]=%f \n", i, dresult);
    sum_deriv += dresult;}
printf("Diff. of Integration:%f \n",sum_deriv);
printf("The integration:   %f \n", presult);
gsl_integration_workspace_free(w);  }
```

In listing 1.3 the algorithmic tangent mode of AD is used to differentiate the integration of a gsl function (listing 1.1), which in the above listing is evaluated

[1] Checkpointing techniques can help keeping the required memory feasible at the expense of additional function evaluations, See [Gri92]. for details.

[2] AD tools are: dco (Derivative Code by Overloading) and dcc (Derivative Code Compiler). In this paper we apply dco as AD tool.

with qags routine. For this purpose, a workspace of size 100000 is defined. The function has n parameters and the boundaries $a, b \in \mathbb{R}$ are dependent to the parameters. Differentiating this function with respect to all parameters with algorithmic tangent, a loop of size n is defined in line 4. After setting the function with its parameters, the integration routine is called in line 11. Furthermore, with every call of the integration routine (i.e. for each $\alpha_i, i = 1, \dots, n$), ν_1 number of iterations will be applied in order to approximate the integral. In algorithmic tangent mode, ν_1 in every call of the integration routine is the same.

Listing 1.4: Algorithmic Adjoint

```
1  gsl_integration_workspace_a1s_type* w =
          gsl_integration_workspace_alloc_a1s_type(100000);
   struct my_f_params<dco::ga1s<double>::type> params;
   params.alpha = vec_alpha;
5  dco::ga1s<double>::global_tape
                 -> register_variable(params.alpha);
   initialise_boundaries(a, b, params.alpha);
   gsl_function_a1s_type F;
   F.function = &func;
10 F.params = &params;
   gsl_integration_qags(&F, a, b, 1e-7, 1e-7, w->limit,
                        w, &result, &error);
   dco::ga1s<double>::set(result, 1, -1);
   dco::ga1s<double>::get(result, presult);
15 dco::ga1s<double>::global_tape->interpret_adjoint();
   for(int ii=0; ii<n; ii++) {
       dco::ga1s<double>::get(params.alpha[ii], aux, -1);
       printf("dI/da[%d]=%f␣\n", ii, aux);
       dresult += aux;}
20 dco::ga1s<double>::global_tape->reset();
   printf("Diff.␣of␣Integration:␣␣␣%f␣\n",dresult);
   printf("The␣integration:%f␣\n", presult);
   gsl_integration_workspace_free(w); }
```

In listing 1.4 the algorithmic adjoint mode of AD is used to differentiate the integration of a gsl function. The same as tangent mode, a workspace of size 100000 is defined, the function has n parameters and the boundaries $a, b \in \mathbb{R}$ are dependent to the parameters. Differentiating this function with respect to all parameters with algorithmic adjoint, the parameters should be registered in tape for backward interpretation. After setting the function with its parameters, the integration routine is called just once in line 11, and like algorithmic tangent mode, by the call of integration routine, ν_1 iterations will be applied in order to approximate the integral. With one interpretation we evaluate the integral as well as the derivative of the integration routine with respect to all parameters.

Computational complexity of n projections with algorithmic tangent and adjoint modes for differentiating the gsl integration routine with ν_1 iterations is $\nu_1 \cdot O(n)$ and $\nu_1 \cdot O(1)$ respectively, and the memory requirement of algorithmic adjoint mode for n projections is $\nu_1 \cdot O(n)$.

4 Symbolic Differentiation of GSL Integration routines

The symbolic differentiation of the integral $I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} f(\boldsymbol{\alpha}, x)dx$ with respect to $\boldsymbol{\alpha}$ is evaluated by computing Equation (4) with $\boldsymbol{\alpha} \in I\!R^n$. In order to evaluate the derivatives in the symbolic mode we apply AD tool, it means the evaluation of ∇a, ∇b and ∇f are done with dco. After computing the derivatives with AD, the integration routine can be called with its original data type *double*. There are two possibilities to compute the derivatives in dco, either with tangent mode AD or with adjoint mode AD. Evaluating the derivatives with tangent mode AD and then integrating the function is straightforward.

Listing 1.5: Function Wrapper Tangent

```
double f_wrapper_t1(double x, void *params){
  struct my_f_params<dco_t1_type> param_alpha =
           *(struct my_f_params<dco_t1_type> *)params;
  dco_t1_type x_active = x;
  dco_t1_type prod;
  dco::gt1s<double>::set
            (param_alpha.alpha[indx], 1., 1);
  prod = func(x_active, &param_alpha);
  double derivative = 0;
  dco::gt1s<double>::get(prod, derivative, 1);
  dco::gt1s<double>::set
             (param_alpha.alpha[indx], 0, 1);
  return derivative;}
```

According to Equation (4), computing the derivatives of the integral with symbolic mode, the differentiation of the function should be passed to the integration routine as integrand instead of the function itself. The above implementation defines the differentiation (with tangent mode) of the function which should be integrated, with respect to one parameter, i.e. $dF\_Tg = (\frac{\partial f(\boldsymbol{\alpha},x)}{\partial \alpha_{indx}})$, where $indx \in [1, n]$. The output is scalar and this function is actually the integrand in the symbolic tangent mode.

Listing 1.6: Symbolic Tangent

```
  gsl_integration_workspacedouble* w =
         gsl_integration_workspace_allocdouble(100000);
  struct my_f_params<double> cont_params;
  for(int i=0; i<n; i++) {
    params.alpha = vec_alpha;
    dco::gt1s<double>::set(params.alpha[i], 1., 1);
    indx = i;
    initialise_boundaries(a, b, params.alpha);
    dco::gt1s<double>::get(a, pa);
    dco::gt1s<double>::get(b, pb);
    dco::gt1s<double>::get(a, da, 1);
    dco::gt1s<double>::get(b, db, 1);
    pre_result = func(pb, &cont_params)*db
               - func(pa, &cont_params)*da;
```

```
15      gsl_functiondouble dF_Tg;
        dF_Tg.function = &f_wrapper_t1;
        dF_Tg.params = &params;
        gsl_integration_qags(&dF_Tg, pa, pb, 1e-7, 1e-7,
                             w->limit, w, &result, &error);
20      dco::gt1s<double>::set(params.alpha[i], 0, 1);
        cont_integral[i] = pre_result + result;
        printf("dI/da[%d]=%f\n", i, cont_integral[i]);
        sum_deriv += cont_integral[i]; }
      printf("Diff. of Integration:%f \n",sum_deriv);
25    gsl_integration_workspace_free(w);   }
```

Listing 1.6 is defined to differentiate the integration of a gsl function with symbolic tangent mode. The function has n parameters and the boundaries a and b are dependent to the parameters. Differentiating this function with respect to all parameters with symbolic tangent, a loop of size n is defined in line 4, which implies the integration routine should be called n times. Furthermore, with every call of the integration routine, ν_2 number of iterations will be applied in order to approximate the integral. In symbolic tangent mode, the number of ν_2 iterations in every call of the integration routine can be different. This is because the integrand is the differentiation of the gsl function $\left(\frac{\partial f(\boldsymbol{\alpha},x)}{\partial \alpha_i}\right)$, $i = 1, \ldots, n$, which can be different for each i.

The differences between this evaluation with the one in listing 1.3 are: in the above implementation, the data type of the variables in the integration routine as well as in gsl function is *double*, whereas in algorithmic tangent they are of $dco :: gt1s < double >:: type$ type, the function which is passed to the integration routine in symbolic tangent is the differentiation of the function which should be passed to the integration routine in algorithmic tangent, additionally $pre\_result = \nabla b f(\boldsymbol{\alpha}, b) - \nabla a f(\boldsymbol{\alpha}, a)$ should be evaluated.

For some routines in gsl, the function which is defined to be integrated differs from the original function which should be integrated. For example, suppose a function which is defined as $f(\boldsymbol{\alpha}, x) = \sum_{i=1}^{i=n} \frac{\sin(\alpha_i \cdot x)}{\alpha_i^2}$. Applying *gsl_integration_qawc* routine (which is an integration routine for integrating the functions with a singularity at c and $c \in (a, b)$) on it, then gsl considers this function as $f(\boldsymbol{\alpha}, x) = \sum_{i=1}^{i=n} \frac{\sin(\alpha_i \cdot x)}{\alpha_i^2 \cdot (x-c)}$ (which in this paper we call the original function), just because of applying *gsl_integration_qawc* on it. Differentiating the integrals with symbolic mode, it should be noticed that for computing $f(a, \boldsymbol{\alpha})$ and $f(b, \boldsymbol{\alpha})$ the original function should be considered as f.

Evaluating the derivatives with adjoint mode AD and then integrating it, is tricky, because in adjoint mode in case of scalar output, with one function call we get the derivative of the output of that function with respect to all inputs.

As it is shown in Equation 4, in order to evaluate the derivatives of the integral with respect to its parameters with symbolic mode, instead of the integrand, the derivative of the integrand with respect to parameters should be integrated.

Differentiating the *gsl_function* with adjoint mode AD in order to calculate ∇f, the output will not be scalar any more, but a vector of size n. According to this reason, a vectorized version of gsl integration routines should be defined. For this purpose, we build a new library e.g. *integration-multidim*, in which the dimension n should be added to the structure of *gsl_function*. Therefore, we define a *gsl_function_vec* as

Listing 1.7: Definition of gsl_function_vec with Parameters

```
struct gsl_function_vec_struct{
  int dim;
  std::vector<double> (* function)
                      int dim, double x, void * params);
  void * params;
};
typedef struct gsl_function_vec_struct gsl_function_vec;
#define GSL_FN_VEC_EVAL(F,x)
           (*((F)->function))((F)->dim,x,(F)->params)
```

Hence, the whole routines, classes, structures and functions should be changed in a way that they can deal with a vector function (and not scalar function as default). In this case, the evaluation of all of the results (i.e. the differentiation of the integral with respect to all parameters) and all of the respective absolute errors are done simultaneously, therefore, *result* and *abserr* (which are outputs) in the integration routines, should be defined as vectors.

The value *n_max* in workspace determines the maximum number of bisections and as result the maximum number of approximations of the results and absolute errors in the interval. The adaptive integration routines in gsl iterate and bisect the integration region until reach to the tolerance. For the cases that we need more iterations (bisections) of the integral region than *n_max*, we get a *GSL_ERROR : the number of iterations was insufficient to reach the tolerance.* By using the adjoint mode AD in the symbolic version the dimension of *rlist* and *elist* in Listing 1.2 should be increased to *n_max* $\times n$ (instead of *n_max*), because the approximation of the integral for our integrand ∇f as well as the absolute error estimates for all $\alpha_i, i = 1, \ldots, n$ will be done at the same time. Allocating *n_max* $\times n$ memory to *result* and *abserr* especially for cases that we need significantly less iterations than *n_max* is not efficient. Therefore, we allocate at first n units of memory to them and with every bisection we increase the size of allocated memory by 1. The dimensions of other *workspace* members stay the same.

In the adaptive routines of gsl integration routines the error estimates are compared and the interval with the largest error is bisected. What should we do now that we have n error estimates for each interval? The answer is, in this paper, we compare the n error estimates and determine the maximum one on each interval and the interval with the largest error would be bisected. It results that, at the end the number of iterations performed by the routine is nearly equal to the largest number of iterations performed by symbolic tangent for each parameter.

Listing 1.8: Function Wrapper Adjoint

```
1  std::vector<double> f_wrapper_a1
2          (int n, double x, void *params) {
3    struct my_f_params<dco_a1m_type> param_alpha =
4                *(struct my_f_params<dco_a1m_type> *)params;
5    ad_mode::tape_t_options options;
6    options.chunksize() = 10*alpha_dim;
7    static ad_mode::tape_t *tape =
8                             ad_mode::tape_t::create(options);
9    dco_a1m_type x_active = x, prod;
10   std::vector<double> deriv(n, 0);
11   tape -> register_variable(param_alpha.alpha);
12   prod = func(x_active, &param_alpha);
13   ad_mode::set(prod, 1., -1);
14   tape->interpret_adjoint();
15   ad_mode::get(param_alpha.alpha, deriv, -1);
16   tape->reset();
17   return deriv;}
```

Listing 1.8 defines the differentiation (with adjoint mode) of the function which should be integrated $dF\_Adj = \nabla f$. This function is actually the integrand in the symbolic adjoint mode. As shown in lines $6-7$ in Listing 1.8, a local tape of size $(10 \times n)$ is defined to store the intermediate variables for the reverse interpretation in order to evaluate dF_Adj.

Listing 1.9: Symbolic Adjoint

```
  gsl_integration_workspace* w =
        gsl_integration_workspace_alloc(100000, n);
  std::vector<double> result(n), error(n);
  struct my_f_params<dco::ga1s<double>::type> params;
  params.alpha = vec_alpha;
  struct my_f_params<double> cont_params;
  dco::ga1s<double>::global_tape ->
                  register_variable(params.alpha);
  initialise_boundaries(a, b, params.alpha);
  dco::ga1s<double>::global_tape->
                       register_output_variable(a);
  dco::ga1s<double>::global_tape->
                       register_output_variable(b);
  dco::ga1s<double>::set(a, 1, -1);
  dco::ga1s<double>::global_tape->interpret_adjoint();
  dco::ga1s<double>::get(a, pa);
  dco::ga1s<double>::get(params.alpha, deriv_a, -1);
  dco::ga1s<double>::global_tape->zero_adjoints();
  dco::ga1s<double>::set(b, 1, -1);
  dco::ga1s<double>::global_tape->interpret_adjoint();
  dco::ga1s<double>::get(b, pb);
  dco::ga1s<double>::get(params.alpha, deriv_b, -1);
  cont_params.alpha = glob_vec_alpha;
```

```
24    aux0 = func(pb, &cont_params);
25    aux1 = func(pa, &cont_params);
26    for(int i=0; i<n; i++)
27        pre_result[i] = aux0*deriv_b[i] - aux1*deriv_a[i];
28    d_params.alpha = vec_alpha_d;
29    gsl_function_vec dF_Adj;
30    dF_Adj.dim = n;
31    dF_Adj.function = &f_wrapper_a1;
32    dF_Adj.params = &d_params;
33    gsl_integration_qags(&dF_Adj, pa, pb, 1e-7, 1e-7,
34                          w->limit, w, result, error);
35    for(int i=0; i<n; i++) {
36        printf(dI/da[%d]=%f \n", i, pre_result[i]+result[i]);
37 ␣␣␣␣␣sum_deriv␣+=␣pre_result[i]␣+␣result[i];␣}
38 ␣␣dco::ga1s<double>::global_tape->reset();
39 ␣␣printf("Diff. of Integration:%f \n",sum_deriv);
40 ␣␣gsl_integration_workspace_free(w);␣␣}
```

Listing 1.9 is defined to differentiate the integration of a gsl function with symbolic adjoint mode. For this purpose, the dimension of the parameters (n) is added to the structure of *workspace_alloc* in order to allocate memory of n to *rlist* and *elist* in the workspace (listing 1.2). The outputs *result* and *abserr* are defined as vectors of size n. The gsl function is here a vector function (listing 1.7). The vector function which should be passed to the integration routine is the adjoint differentiation of the integrand (listing 1.8). After setting the function with its parameters and computing $pre\_result = \nabla b f(\boldsymbol{\alpha}, b) - \nabla a f(\boldsymbol{\alpha}, a)$, the integration routine is called just once in line 33, and by the call of integration routine, ν_2 iterations will be applied in order to approximate the integral. In the symbolic adjoint mode, the number of ν_2 iterations is nearly equal to the maximum ν_2 number of iterations in symbolic tangent.

Computational complexity of n projections with symbolic tangent and adjoint modes for differentiating the gsl integration routine (e.g. qags) with ν_2 iterations is $\nu_2 \cdot O(n)$ and $\nu_2 \cdot O(1)$ respectively. The memory requirement of symbolic adjoint mode for n projections is $O(n)$, which contains the memory requirement for evaluating ∇f, that in this paper is defined to be $(10 \times n)$, and the memory requirement of computing ∇a and ∇b, that is also $O(n)$.

5 Test Cases

This chapter describes and compares routines for performing numerical integration (quadrature) of a function with multi dimensional parameters and the differentiation of the integration with different methods, i.e. algorithmic tangent/adjoint and symbolic tangent/adjoint. It is important to choose a function as test case, in which the corresponding integration routine is suitable for that test case and also the same integration routine is suitable for the differentiation of that function, because in this paper we use the same integration routine for both symbolic and algorithmic modes. The duration of the computation depends strongly on the number of iterations performed by the integration routine and the number of iteration depends on the integrand and the specified accuracy. As mentioned in the previous section, the number of iterations for algorithmic and symbolic computation can differ, because in algorithmic version, the integrand is the function, however, in symbolic mode, the integrand is the derivative of the function with respect to its parameters. Furthermore, the number of iterations in symbolic tangent differentiation can be different for each parameter, but with applying symbolic adjoint, we have just one number of iterations, which is nearly the same as maximum number of iterations applied by symbolic tangent. In this section, in case of illustrating the number of iterations with symbolic mode, we consider the number of iterations applied by symbolic adjoint .

All of the following measurements are done on a machine with 2x Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz (2x 6 Cores (12 Threads)), 128 GB RAM.

1. **QAG adaptive integration**: The QAG algorithm is a simple adaptive integration procedure. The adaptive functions apply an integration rule adaptively until an estimate of the integral of f over (a, b) is achieved within the desired absolute and relative error limits, *epsabs* and *epsrel*. The function returns the final approximation, *result*, and an estimate of the absolute error, *abserr*.

   As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \frac{\cos(\alpha_i^3 \cdot x)}{x} dx \quad .$$

   where $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \dots, n$ using qag routine. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead is shown in Figure 1 and in Table 1 the memory requirement as well as number of iterations are illustrated.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff
5	0.43	0.16	0.4	0.16	0.43
10	1.17	0.56	2.11	0.44	2.25
15	3.34	0.8	6.19	0.84	6.55
20	8.16	2.08	21.52	2.21	23.55
50	91.79	10.08	266.4	10.8	286.12
80	-	31.85	-	34.21	-

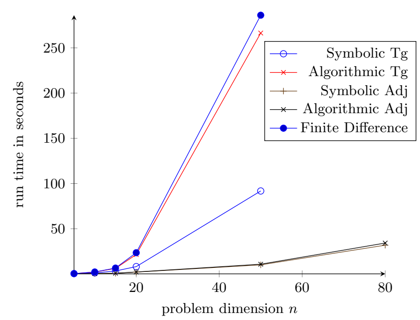

Fig. 1: Run time overhead in seconds for qag routine. Missing values indicate failure to converge within 300 seconds.

   In this test case, absolute error is set as well as relative error to 10^{-7}. To reach this accuracy, e.g. for $n = 10$, the number of iterations applied by symbolic and algorithmic is 2041 and 1523 respectively. As illustrated in Figure 1, evaluating derivatives with adjoint modes is considerably more efficient in terms

n	Symbolic Adj	Symbolic ν	Algorithmic Adj	Algorithmic ν
5	0.001	1024	97.02	1024
10	0.002	2041	250.62	1523
15	0.003	2048	479.96	2048
20	0.004	4096	1233.82	4056
50	0.01	8192	5923.07	8192
80	0.018	16384	18647.4	16330

Table 1: Memory Requirement in MB and number of iterations (ν) for qag routine.

of runtime than applying tangent modes. For this function with this accuracy, symbolic tangent is faster than algorithmic tangent, whereas symbolic and algorithmic adjoint have nearly the same runtime overhead. However, the memory requirement of algorithmic adjoint is significantly higher than the one for symbolic adjoint.

(a) QAG $n = 15$ (b) QAG $n = 50$

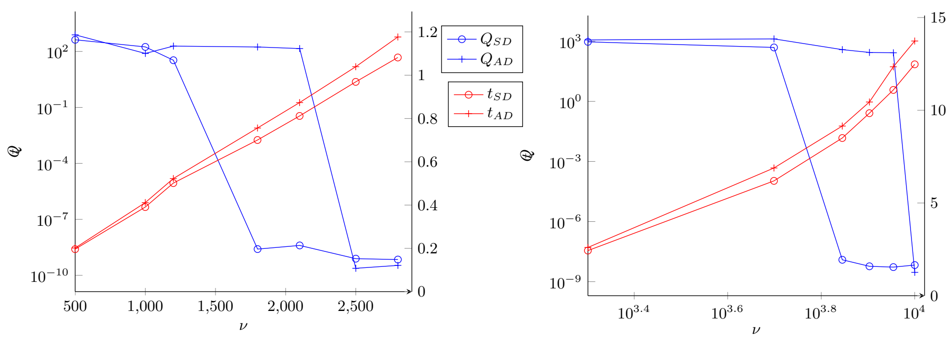

Fig. 2: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qag routine using different approaches to differentiation.

Figure 2 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different number of iterations ν using different differentiation methods in the integration of the reference problem with qag routine for $n = 15$ and $n = 50$. In this section the discrepancy between the adjoints computed in νth iteration of the integration routine with its value in the previous iteration yields:

$$Q = \|\boldsymbol{\alpha}_{(1)\nu_j} - \boldsymbol{\alpha}_{(1)\nu_{(j-1)}}\| \quad , \quad j \in (1, n\_max) \quad . \tag{5}$$

Suppose the convergence $\delta = 10^{-8}$, Figure 2 shows that to reach this accuracy, algorithmic adjoint requires $\nu = 2500$ and $\nu = 10000$ iterations whereas symbolic adjoint needs $\nu = 1800$ and 8000 iterations for $n = 15$ and $n = 50$ respectively. The behaviour of symbolic and algorithmic adjoints in terms of runtime overhead is nearly the same, however symbolic adjoint is a bit faster.

2. **QAGS adaptive integration with singularities**:The presence of an integrable singularity in the integration region causes an adaptive routine to concentrate new subintervals around the singularity. As the subintervals decrease

in size the successive approximations to the integral converge in a limiting fashion. This approach to the limit can be accelerated using an extrapolation procedure. The QAGS algorithm combines adaptive bisection with the Wynn epsilon-algorithm to speed up the integration of many types of integrable singularities.

As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \frac{\alpha_i^3 \cdot \sin(x)}{x} dx \quad .$$

where $a(\boldsymbol{\alpha}) = -\sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \dots, n$ using qags routine. This function has a singularity in $x = 0$. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 3 and Table 3 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
10	0.01	0.01	0.01	0.004	0.01
50	0.4	0.04	0.31	0.04	0.28
100	2.3	0.14	2.01	0.19	2.05
300	51.45	1.19	50.49	1.63	47.05
500	203.88	2.81	191.17	3.8	180.11
1000	-	11.74	-	15.1	-
1500	-	32.66	-	42.47	-

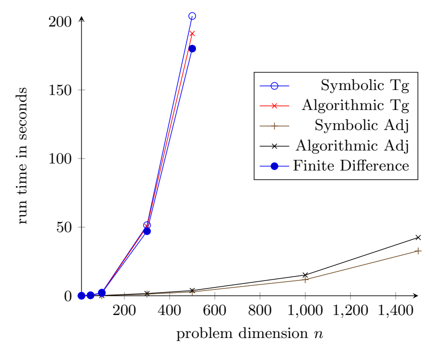

Fig. 3: Run time overhead in seconds for qags routine. Missing values indicate failure to converge within 300 seconds.

	Symbolic		Algorithmic	
n	Adj	ν	Adj	ν
10	0.002	31	1.75	31
50	0.01	128	31.82	128
100	0.02	254	124.18	254
300	0.07	722	1047.01	722
500	0.1	1022	2464.34	1022
1000	0.2	2043	9836.12	2043
1500	0.3	3880	28006	3880

Table 2: Memory Requirement in MB and number of iterations (ν) for qags routine.

In this test case, absolute error is set as well as relative error to 10^{-7}. To reach this accuracy, e.g. for $n = 100$, the number of iterations for both symbolic and algorithmic is 254. As illustrated in Figure 3, for this function with this accuracy, algorithmic tangent is faster than symbolic tangent, whereas symbolic adjoint is faster than algorithmic adjoint. Furthermore, the memory requirement of algorithmic adjoint is considerably higher than the memory requirement of the symbolic adjoint.

Figure 4 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different νs using different differentiation methods in the integration of the reference problem with qags routine and shows that symbolic adjoint converges faster in comparison with algorithmic adjoint, e.g. for $\delta = 10^{-10}$ and $n = 300$, symbolic adjoint needs $\nu = 722$ and algorithmic adjoint requires $\nu = 1022$ iterations. Furthermore, the time spent by symbolic adjoint is less than time spent by algorithmic adjoint. By increasing n and ν, the difference between duration of computation by adjoint algorithmic and symbolic becomes larger.

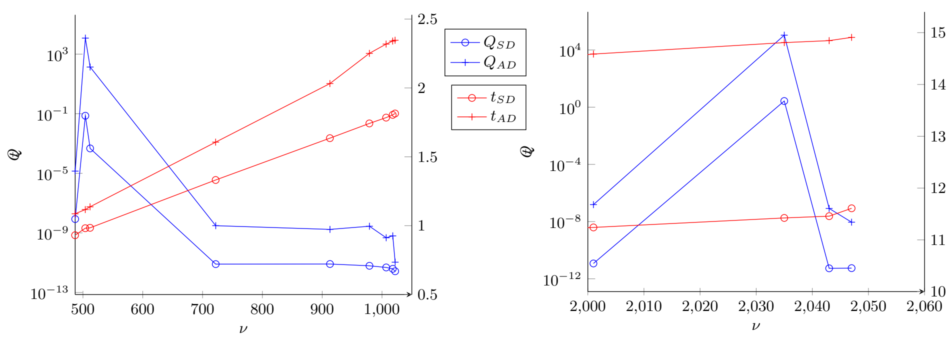

Fig. 4: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qags routine using different approaches to differentiation.

As mentioned before, we should take care of choosing right integrand in order to be able to apply the same integration routine on the corresponding integrand for both symbolic and algorithmic differentiation. For example, suppose a function $f(\boldsymbol{\alpha}, x) = \sum_{i=1}^{n} \frac{\sin(\alpha_i^3 \cdot x)}{x}$ which has singularity at $x = 0$, therefore the integration routine qags can be applied on it, but the differentiation of the function $\nabla f = \sum_{i=1}^{n} 2\alpha_i^2 \cos(\alpha_i^3 \cdot x)$ which will be the integrand by using symbolic differentiation has no singularity at $x = 0$, therefore applying qags routine on it is not efficient.

3. **QAGI adaptive integration on infinite intervals**: This algorithm uses the QAGS algorithm, which computes the integral of the function f over the infinite interval $(-\inf, +\inf)$. The integral is mapped onto the semi-open interval $(0, 1]$ using the transformation $x = (1 - t)/t$.

4. **QAGIU adaptive integration with infinite upper boundary**: This algorithm uses the QAGS algorithm, which computes the integral of the function f over the semi-infinite interval $(a, +\inf)$. The integral is mapped onto the semi-open interval $(0, 1]$ using the transformation $x = a + (1 - t)/t$.

5. **QAGIL adaptive integration with infinite upper boundary**: This algorithm uses the QAGS algorithm, which computes the integral of the func-

tion f over the semi-infinite interval $(-\inf, b)$. The integral is mapped onto the semi-open interval $(0, 1]$ using the transformation $x = b - (1-t)/t$.

6. **QAWC adaptive integration with one singularity at x=c**: This function computes the Cauchy principal value of the integral of f over (a, b), with a singularity at c, $I = \int_a^b dx f(x)/(x-c)$. The adaptive bisection algorithm of QAG is used, with modifications to ensure that subdivisions do not occur at the singular point $x = c$. When a subinterval contains the point $x = c$ or is close to it then a special 25-point modified Clenshaw-Curtis rule is used to control the singularity. Further away from the singularity the algorithm uses an ordinary 15-point Gauss-Kronrod integration rule.
This routine is used by integrands with weight functions and for evaluation of integrals with this method, table of chebyshev moments in every iteration of the integration routine should be computed. For this purpose, there exist two variables: Cheb12 and Cheb24 of size 13 and 25 respectively. In the vectorized version of gsl, the size of Cheb12 and Cheb24 should be increased by factor of n in order to make the simultaneous computation for all parameters possible.

As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \frac{\sin(\alpha_i^3 \cdot x)}{(x-c)} dx \quad ,$$

where $c \in (a, b)$, $a(\boldsymbol{\alpha}) = -\sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using qawc routine. This function has a singularity in $c \in (a, b)$ in both symbolic and algorithmic versions. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 5 and Table 5 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
5	1.29	0.76	2.05	0.73	2.84
10	4	1.34	7.98	1.36	9.97
20	28.57	8.46	101.23	7.72	133.81
30	67.27	15.29	273.66	14.85	390.26
35	99.18	17.28	361.85	17.17	510.17

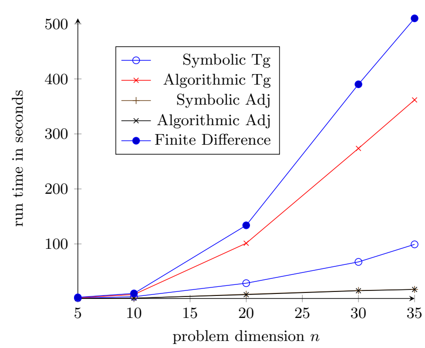

Fig. 5: Run time overhead in seconds for qawc routine.

In this test case, absolute error is set as well as relative error to 10^{-7}. To reach this accuracy, e.g. for $n = 10$, the number of iterations for symbolic and algorithmic is 16614 and 17906 respectively. As illustrated in Figure 5, for this function with this accuracy, symbolic tangent is faster than algorithmic tangent, whereas the behaviour of adjoint symbolic and algorithmic in terms

n	Symbolic Adj	Symbolic ν	Algorithmic Adj	Algorithmic ν
5	0.001	15860	368.34	15693
10	0.002	16614	686.17	17906
20	0.004	48034	3276.37	48109
30	0.007	64540	6269.02	64065
35	0.008	66374	7460.27	66198

Table 3: Memory Requirement in MB and number of iterations (ν) for qawc routine.

of runtime is the same. However, the memory requirement of algorithmic adjoint is considerably higher than the memory requirement of the symbolic adjoint.

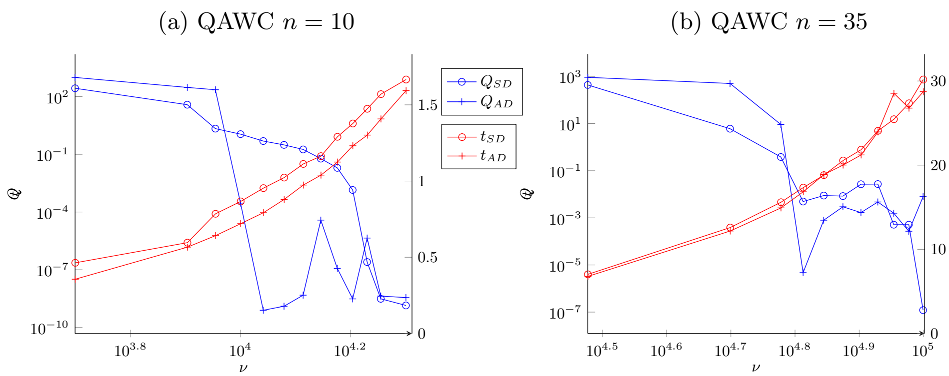

Fig. 6: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qawc routine using different approaches to differentiation.

Figure 6 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different νs using different differentiation methods in the integration of the reference problem with qawc routine and shows that at first ($\nu < 11000$ and $\nu < 65000$ for $n = 15$ and $n = 35$ respectively) symbolic adjoint converges faster in comparison with algorithmic adjoint, but after that the convergence of algorithmic adjoint gets more speed. For $n = 10$ adjoint algorithmic spends less time, but for $n = 35$ the time spent for both methods is nearly the same.

7. **QAWS adaptive integration for functions with singular endpoints**: The QAWS algorithm is designed for integrands with algebraic-logarithmic singularities at the end-points of an integration region. In order to work efficiently the algorithm requires a precomputed table of Chebyshev moments. The adaptive bisection algorithm of QAG is used. When a subinterval contains one of the endpoints then a special 25-point modified Clenshaw-Curtis rule is used to control the singularities. For subintervals which do not include the endpoints an ordinary 15-point Gauss-Kronrod integration rule is used. This routine is used by integrands with weight functions and for evaluation of integrals with this method, table of chebyshev moments in every iteration of the integration routine should be computed.

As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a=0}^{b=1} \sum_{i=1}^{n} \frac{\cos(\alpha_i^2) \cdot (\log(\alpha_i \cdot x))^2}{(1-x)^2 \cdot \alpha_i^2} dx \quad .$$

with respect to its parameters $\alpha_i > 0, i = 1, \dots, n$ using qaws routine. This function is singular in endpoints a and b, therefore, the endpoints should not depend on α, because in case of dependent boundaries in symbolic differentiation of integrals, $f(a, \alpha)$ and $f(b, \alpha)$ should be computed, which in this case do not exit because of the singularity. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 7 and Table 4 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
10	0.05	0.01	0.04	0.01	0.04
100	2.68	0.06	2.58	0.05	3.2
500	71.04	0.29	72.29	0.28	87.62
700	138.68	0.41	141.5	0.40	172.37
1000	-	0.74	-	0.6	-
5000	-	3.19	-	2.88	-
10000	-	6.27	-	5.77	-
20000	-	12.53	-	11.6	-
50000	-	31.83	-	28.92	-
100000	-	65.27	-	57.84	-

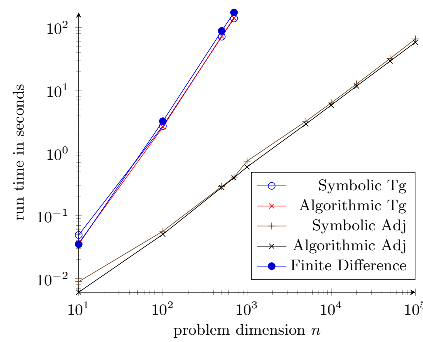

Fig. 7: Run time overhead in seconds for qaws routine. Missing values indicate failure to converge within 300 seconds.

	Symbolic		Algorithmic	
n	Adj	ν	Adj	ν
10	0.002	49	3.61	46
100	0.02	50	31.97	49
500	0.09	50	157.06	49
700	0.12	50	219.6	49
1000	0.18	50	313.41	49
5000	0.89	50	1564.24	49
10000	1.75	50	3127.77	49
20000	3.51	50	6254.83	49
50000	8.77	50	15636	49
100000	17.55	50	31271.4	49

Table 4: Memory Requirement in MB and number of iterations (ν) for qaws routine.

In this test case, absolute error is set as well as relative error to 10^{-12}. To reach this accuracy, e.g. for $n = 500$, the number of iterations for symbolic and algorithmic is 50 and 49 respectively. As illustrated in Figure 7, for this function with this accuracy, for small n tangent methods as well as adjoint methods have the same runtime overhead, but by increasing the size n, algorithmic adjoint spends less time compared to other differentiation methods. The memory requirement of algorithmic adjoint is significantly higher than the memory requirement of symbolic adjoint.

(a) QAWS $n = 100$ (b) QAWS $n = 1000$

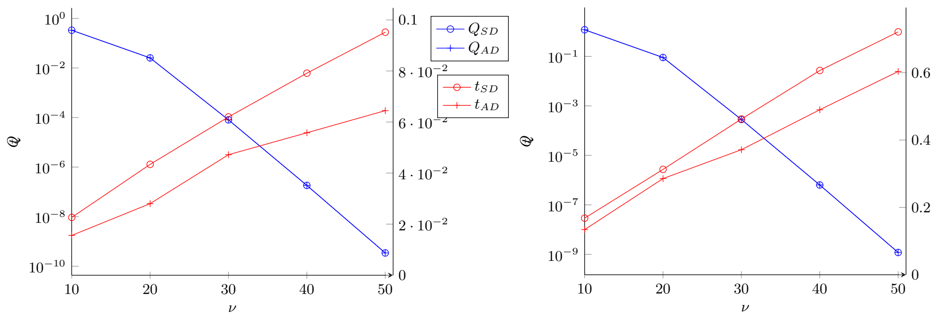

Fig. 8: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qaws routine using different approaches to differentiation.

Figure 8 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different νs using different differentiation methods in the integration of the reference problem with qaws routine. Because of independence of the boundaries to parameters, in Equation (4) we have $\nabla b = 0$ and $\nabla a = 0$. It results the same behaviour in terms of convergence for both adjoint algorithmic and symbolic methods. However, because of computation of the Chebyshev table of moments in qaws and increasing the size of it in the symbolic adjoint mode, the symbolic adjoint methods spends more time than algorithmic adjoint method. The difference of duration for symbolic and algorithmic for $n = 100$ is ≈ 0.03 and for $n = 1000$ is ≈ 0.12 seconds.

8. **QAWO adaptive integration for oscillatory functions**: This algorithm is designed for integrands with an oscillatory factor, $\sin(\omega x)$ or $\cos(\omega x)$. In order to work efficiently the algorithm requires a table of Chebyshev moments which must be pre-computed. Those subintervals with *large* widths where $d\omega > 4$ are computed using a 25-point Clenshaw-Curtis integration rule, which handles the oscillatory behavior. Subintervals with a *small* widths where $d\omega < 4$ are computed using a 15-point Gauss-Kronrod integration.
This routine is used by integrands with weight functions and for evaluation of integrals with this method, table of chebyshev moments in every iteration of the integration routine should be computed.
QAWF routine (see below) uses QAWO in the computation of integrals.

9. **QAGP adaptive integration with known singular points**: This function applies the adaptive integration algorithm QAGS taking account of the user-supplied locations of singular points. The array *pts* of length *npts* should contain the endpoints of the integration ranges defined by the integration region and locations of the singularities. If you know the locations of the singular points in the integration region then this routine will be faster than QAGS.

As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \alpha_i^3 \cdot x^3 \log\left(\frac{|(x^3 - p_1^3)\cdot(x^2 - p_2^2)|}{\alpha_i + 1}\right) dx \quad .$$

where $p_1, p_2 \in (a,b)$, $p_1 < p_2$, $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = 4 * \sum_{i=1}^{n} \alpha_i$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using qagp routine. This function is singular in $x = p_1$ and $x = \pm p_2$, however, $x = -p_2$ is not in our integration region. Therefore, we have 2 singular points.

```
std::vector<double> pts(4, 0);
pts[0] = a;
pts[1] = p1;
pts[2] = p2;
pts[3] = b;
gsl_integration_qagp (&f, pts, n, 1e-7, 1e-7, w->limit,
                      w, &result, &abserr) ;
```

Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 9 and Table 5 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
50	0.22	0.01	0.19	0.01	0.26
100	0.75	0.01	0.62	0.01	0.94
1000	62.74	0.14	61.93	0.14	92.59
5000	1715.54	0.72	1569.16	0.74	2321.33
10000	-	1.54	-	1.47	-
50000	-	7.27	-	7.31	-
100000	-	15.31	-	14.68	-
500000	-	81.04	-	73.85	-
1000000	-	167.91	-	156.56	-

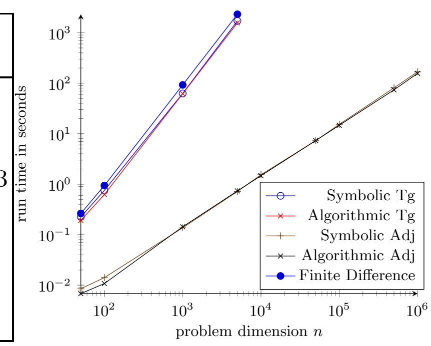

Fig. 9: Run time overhead in seconds for qagp routine. Missing values indicate failure to converge within 3000 seconds.

In this test case, absolute error is set as well as relative error to 10^{-7}. To reach this accuracy, e.g. for $n = 5000$, the number of iterations for both symbolic and algorithmic is 21. As illustrated in Figure 9, for this function with this accuracy, algorithmic and symbolic adjoint methods have nearly the same runtime overhead, but the memory requirement of algorithmic adjoint is significantly higher.
Figure 10 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different νs using different differentiation methods in the integration of the reference problem with qagp routine. It shows that, for $n = 100$ and different number of iterations, algorithmic adjoint spends less time, however algorithmic and symbolic adjoint have nearly the same runtime behaviour for $n = 1000$. The convergence of both methods are nearly the same.

	Symbolic		Algorithmic	
n	Adj	ν	Adj	ν
50	0.01	20	4.88	21
100	0.02	21	9.56	21
1000	0.21	21	93.72	21
5000	1.03	21	467.81	21
10000	2.06	21	935.42	21
50000	10.3	21	4676.28	21
100000	20.6	21	9352.33	21
500000	103	21	46760.8	21
1000000	206	21	93521.4	21

Table 5: Memory Requirement in MB and number of iterations (ν) for qagp routine.

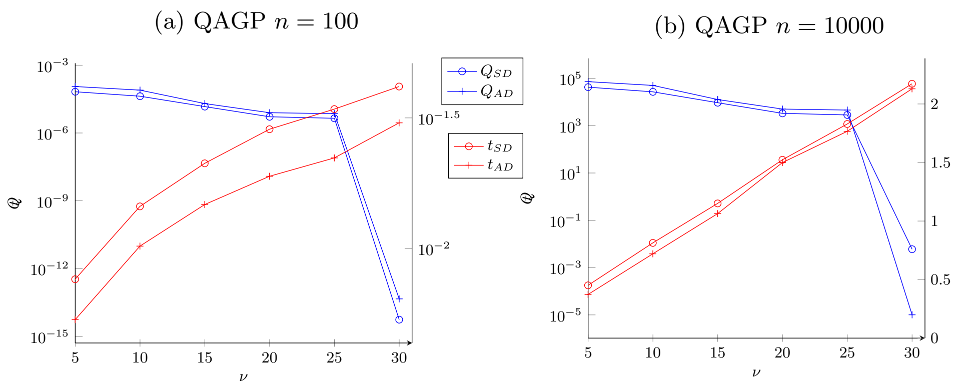

Fig. 10: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qagp routine using different approaches to differentiation.

10. **QNG nonadaptive Gauss-Kronrod integration**: The QNG algorithm is a non-adaptive procedure which uses fixed Gauss-Kronrod-Patterson abscissae to sample the integrand at a maximum of 87 points. It is provided for fast integration of smooth functions.

    As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n-1} x^{(\alpha_i+\alpha_{i+1})} \cdot \sin(\frac{\alpha_i}{x})$$

    where $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = 3 \cdot \sum_{i=1}^{n} \alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using qng routine. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 11 and Table 6 respectively.
    In this test case, the absolute error as well as relative error is set to 10^{-7}. As illustrated in Figure 11, for this function with this accuracy, algorithmic tangent has nearly the same behaviour as symbolic tangent, symbolic adjoint spends less time than algorithmic adjoint for large ns. This routine is not adaptive and therefore it has always the same number of iterations. Memory requirement of algorithmic adjoint is significantly higher than memory requirement of symbolic adjoint.

n	Symbolic Tg	Symbolic Adj	Algorithmic Tg	Algorithmic Adj	Finite Diff.
10	0.003	0.0004	0.003	0.0004	0.002
100	0.2	0.002	0.16	0.002	0.1
1000	13.52	0.01	13.54	0.02	8.08
5000	339.82	0.08	336.3	0.1	205.49
10000	-	0.21	-	0.21	-
100000	-	1.61	-	1.79	-
1000000	-	15.46	-	17.91	-
5000000	-	78.43	-	90.26	-

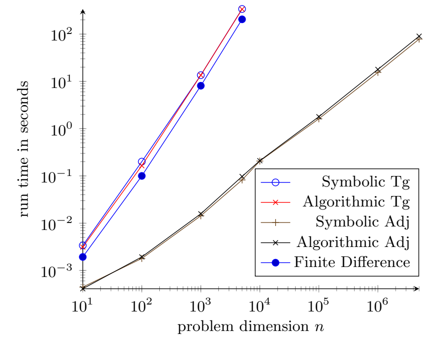

Fig. 11: Run time overhead in seconds for qng routine. Missing values indicate failure to converge within 500 seconds.

n	Symbolic Adj	Algorithmic Adj
10	0.002	0.05
100	0.02	0.51
1000	0.22	5.05
5000	1.11	25.26
10000	2.21	50.51
100000	22.12	505.07
1000000	221.25	5050.68
5000000	1106.26	25253.4

Table 6: Memory Requirement in MB for qng routine.

11. **QAWF adaptive integration for Fourier integrals**: This function attempts to compute a Fourier integral of the function f over the semi-infinite interval $[a, +\infty)$. The subintervals and their results are stored in the memory provided by workspace. The integration over each subinterval uses the memory provided by *cycle_workspace* as workspace for the QAWO algorithm.

    As case study, we consider evaluating the differentiation of the integral

    $$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{\infty} \sum_{i=1}^{n} \frac{\alpha_i^2}{(\alpha_i^2+1)\cdot\sqrt{x+\alpha_i}}$$

    where $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using qawf routine. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 12 and Table 7 respectively.
    In this test case, the absolute error is set to 10^{-8}. To reach this accuracy, e.g. for $n = 5000$, the number of iterations for symbolic and algorithmic is 9 and 11 respectively. As illustrated in Figure 12, for this function with this accuracy, symbolic tangent is faster than algorithmic tangent, algorithmic adjoint and symbolic adjoint have the same behaviour.
    Figure 13 illustrates the convergence (blue lines) and run time (red lines) for computing adjoints with different number of iterations ν using different differentiation methods in the integration of the reference problem with qawf routine for $n = 100$ and $n = 100000$. For both cases the convergence of

n	Symbolic Tg	Symbolic Adj	Algorithmic Tg	Algorithmic Adj	Finite Diff.
10	0.01	0.002	0.01	0.01	0.004
50	0.07	0.004	0.05	0.01	0.04
100	0.2	0.01	0.18	0.01	0.14
500	2.95	0.02	3.62	0.03	2.8
1000	11.31	0.05	13.42	0.05	10.34
5000	272.4	0.22	308.36	0.25	239.56
10000	-	0.45	-	0.5	-
50000	-	1.89	-	2.11	-
100000	-	3.82	-	4.18	-
500000	-	16.42	-	16.45	-
1000000	-	33.07	-	33.96	-

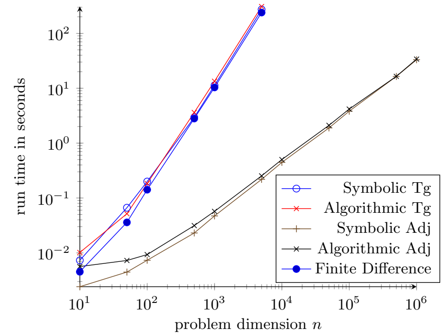

Fig. 12: Run time overhead in seconds for qawf routine. Missing values indicate failure to converge within 400 seconds.

n	Symbolic Adj	Symbolic ν	Algorithmic Adj	Algorithmic ν
10	0.002	10	0.55	9
50	0.01	10	2.29	10
100	0.02	11	4.74	11
500	0.1	10	22.93	11
1000	0.2	10	42.41	12
5000	0.99	9	195.1	11
10000	1.98	9	390.03	11
50000	9.92	7	1625.2	9
100000	19.84	7	3250.27	9
500000	99.18	5	13008.2	7
1000000	198.36	5	26016.4	7

Table 7: Memory Requirement in MB and number of iterations (ν) for qawf routine.

symbolic is faster than algorithmic. Furthermore, symbolic adjoint takes less time than algorithmic adjoint for $n = 100$, however, it is not the case for $n = 100000$. This is because, qawf uses qawo in its implementation and in qawo table of Chebyshev moments should be computed.

(a) QAWF $n = 100$ (b) QAWF $n = 100000$

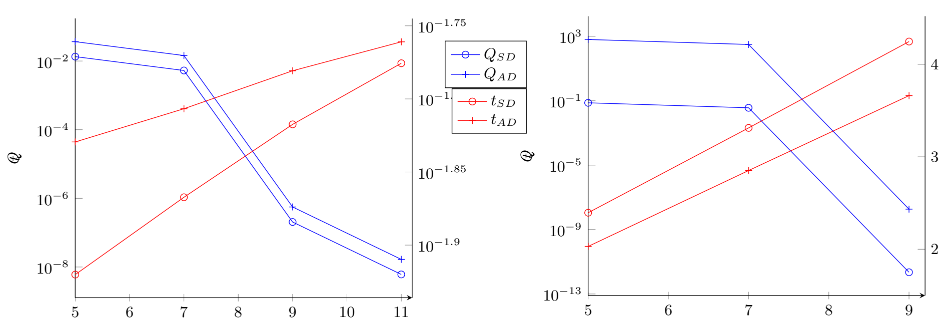

Fig. 13: Discrepancy and run time (in seconds) for evaluation of adjoints of the integration of the reference problem with qawf routine using different approaches to differentiation.

12. **GLFIXED Gauss-Legendre integration**: The fixed-order Gauss-Legendre integration routines are provided for fast integration of smooth functions with known polynomial order. The m-point Gauss-Legendre rule is exact for polynomials of order $2 \cdot m - 1$ or less. Unlike other numerical integration routines within the library, these routines do not accept absolute or relative error bounds.

   As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \frac{20}{2 \cdot m - 1} \cdot \left(\left(\frac{3\alpha_i^2 x}{100} \right)^{2 \cdot m - 1} - \left(\frac{\alpha_i x}{10} \right)^{2 \cdot m - 1} \right)$$

   where $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = \sum_{i=1}^{n} 3\alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using glfixed routine. In this test case, we set $m = 10$. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 14 and Table 8 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
10	0.001	0.0002	0.001	0.0001	0.0001
100	0.07	0.001	0.05	0.001	0.004
1000	3.68	0.01	3.62	0.005	0.34
5000	88.95	0.03	92.16	0.03	9.31
10000	360.91	0.05	371.51	0.05	34.31
100000	-	0.65	-	0.54	-
1000000	-	5.59	-	5.31	-
5000000	-	27.95	-	26.93	-

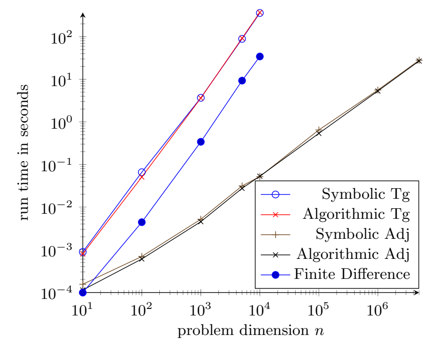

Fig. 14: Run time overhead in seconds for glfixed routine. Missing values indicate failure to converge within 500 seconds.

	Symbolic	Algorithmic
n	Adj	Adj
10	0.003	0.02
100	0.03	0.2
1000	0.3	2.04
5000	1.49	10.19
10000	2.98	20.37
100000	29.75	203.71
1000000	297.55	2037.05
5000000	1487.73	10185.3

Table 8: Memory Requirement in MB for glfixed routine.

As illustrated in Figure 14, for this function with this accuracy, symbolic tangent is faster than algorithmic tangent for large ns, algorithmic and symbolic adjoint have the same behaviour in terms of runtime, however, algorithmic adjoint requires higher memory requirement.

13. **CQUAD doubly-adaptive integration**: CQUAD is a new doubly-adaptive general-purpose quadrature routine which can handle most types of singularities, non-numerical function values such as Inf or NaN, as well as some divergent integrals. It generally requires more function evaluations than the integration routines in QUADPACK, yet fails less often for difficult integrands. The underlying algorithm uses a doubly-adaptive scheme in which Clenshaw-Curtis quadrature rules of increasing degree are used to compute the integral in each interval. The L_2-norm of the difference between the underlying interpolatory polynomials of two successive rules is used as an error estimate. The interval is subdivided if the difference between two successive rules is too large or a rule of maximum degree has been reached.

    As case study, we consider evaluating the differentiation of the integral

$$I(\boldsymbol{\alpha}) = \int_{a(\boldsymbol{\alpha})}^{b(\boldsymbol{\alpha})} \sum_{i=1}^{n} \frac{\alpha_i^2}{\sqrt{(\alpha_i^2+1)\cdot x}}$$

    where $a(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i$ and $b(\boldsymbol{\alpha}) = \sum_{i=1}^{n} 3\alpha_i^2$ with respect to its parameters $\alpha_i > 0, i = 1, \ldots, n$ using cquad routine. Differentiating this integral with algorithmic and symbolic tangent and adjoint for different dimensions of α, the computational overhead as well as memory requirement are shown in Figure 15 and Table 9 respectively.

	Symbolic		Algorithmic		Finite
n	Tg	Adj	Tg	Adj	Diff.
10	0.0005	0.0002	0.0004	0.0002	0.0004
100	0.03	0.001	0.02	0.001	0.02
1000	1.51	0.007	1.34	0.003	1.06
5000	36.44	0.04	33.48	0.02	26.67
10000	157.85	0.07	134.93	0.05	107.57
100000	-	0.89	-	0.48	-
1000000	-	8.96	-	4.8	-

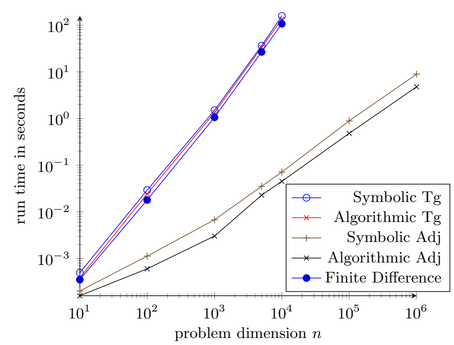

Fig. 15: Run time overhead in seconds for cquad routine. Missing values indicate failure to converge within 300 seconds.

	Symbolic	Algorithmic
n	Adj	Adj
10	0.002	0.12
100	0.02	0.47
1000	0.24	3.99
5000	1.18	19.61
10000	2.37	39.14
100000	23.65	390.71
1000000	236.51	3906.34

Table 9: Memory Requirement in MB for cquad routine.

In this test case, the absolute and relative error is set to 10^{-12}. In the implementation of cquad routine with symbolic adjoint, additional operations should be done, in order to compute the n results and errors simultaneously. As illustrated in Figure 15, for this function with this accuracy, algorithmic tangent is faster than symbolic tangent, algorithmic adjoint has less runtime in comparison to other methods, however, algorithmic adjoint requires higher memory requirement.

6 Summary

In this paper we discussed algorithmic and symbolic differentiation of integrals with multi-dimensional parameters. The run time and memory overhead for algorithmic and symbolic approaches to the differentiation of the integrals with ν_1 and ν_2 (e.g. qags) iterations is shown in Table 10.

	Symbolic		Algorithmic	
	Tangent	Adjoint	Tangent	Adjoint
Memory	$O(n)$	$O(n)$	$O(n)$	$\nu_1 \cdot O(n)$
Run Time	$\nu_2 \cdot O(n)$	$\nu_2 \cdot O(1)$	$\nu_1 \cdot O(n)$	$\nu_1 \cdot O(1)$

Table 10: Computational complexity and memory requirement of n projections of the integral with algorithmic/symbolic tangent and adjoint modes of differentiation for ν_1 algorithmic and ν_2 symbolic (e.g. qags) iterations applied to the integrand/differentiation of the integrand with n parameters.

Computing the differentiation of the integral with symbolic tangent and algorithmic modes, the integration routine stays the same and just the data types should be changed. As shown in Section 1, in symbolic tangent/adjoint mode, the differentiation of the function should be passed to the integration routine. Evaluating the derivative of the integral with symbolic adjoint, the differentiation of the function is not scalar any more, but a vector. This should be considered in every function and routine of the integration and this is the reason to build vectorized functions and integration routines in gsl in order to make the results and errors be evaluated simultaneously.

In Section 5, we observe the differences between algorithmic and symbolic in evaluation of the derivative of the integrals with different routines. Note that, the number of iterations in symbolic and algorithmic is not always the same, because the algorithmic one integrates the function, whereas the symbolic one integrates the differentiation of the function with respect to parameters. Furthermore, in the symbolic tangent version, the numbers of iterations are different (or at least should not be the same) in every projection, due to integrating the differentiation of the function and having different values for each parameter, however, in algorithmic tangent all of the projections are done with the same number of iterations. Evaluation of tangents of the integrals with symbolic and algorithmic modes has nearly the same runtime overhead. This is also the case in evaluation of adjoints. Applying adjoint differentiation of the integrals is better alternative than applying the tangent one, because of independence of the computational cost to the n in adjoint mode.

It is also shown that the runtime overhead of symbolic and algorithmic modes depends on the problem size n and number of iterations ν. In computation of the adjoints, for small n and ν algorithmic version is slightly faster, because of additional computation of $f(\boldsymbol{\alpha}, b(\boldsymbol{\alpha}))\nabla b - f(\boldsymbol{\alpha}, a(\boldsymbol{\alpha}))\nabla a$ in the symbolic version. By increasing ν, the symbolic mode will be faster, because the algorithmic one should go through the algorithm line by line ν times and register the active variables for reverse interpretation and compute the derivatives. This requires memory as well as runtime. Increasing n and having the same ν, symbolic and algorithmic would have the same runtime (Figure 10), except the cases that the table of Chebyshev moments should be computed, in this case because of increasing the dimensions Cheb12 and Cheb24 in every iteration with factor of n in vectorized gsl, the symbolic version requires more runtime (Figure 13).

Furthermore, in most of the integration routines the convergence of symbolic is faster than algorithmic one. Additionally, the memory requirement of algorithmic adjoint is significantly higher than the memory requirement of the symbolic adjoint.

References

[Fla73] H. Flanders. Differentiation under the integral sign. *The American Mathematical Monthly*, 80(6):615–627, Jun. - Jul., 1973.

[GDT+09] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. *GNU Scientific Library Refrence Manual*. Network Theory Ltd, Berlin, third edition, 2009.

[Gri92] A. Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1:35–54, 1992.

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008.

[Nau12] U. Naumann. *The Art of Differentiating Computer Programs. An Introduction to Algorithmic differentiation*. Number 24 in Software, Environments, and Tools. SIAM, Philadelphia, PA, 2012.

Aachener Informatik-Berichte

This is the list of all technical reports since 1987. To obtain copies of reports please consult

http://aib.informatik.rwth-aachen.de/

or send your request to:

Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen,
Email: biblio@informatik.rwth-aachen.de

1987-01 * Fachgruppe Informatik: Jahresbericht 1986
1987-02 * David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03 * Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04 * Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05 * Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06 * Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07 * Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08 * Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09 * Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10 * Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11 * Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12 J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01 * Gabriele Esser, Johannes Rückert, Frank Wagner Gesellschaftliche Aspekte der Informatik
1988-02 * Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03 * Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04 * Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05 * Peter Martini: Performance Analysis of Multiple Token Rings
1988-06 * Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07 * Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08 * Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09 * W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs

1988-10 * Kai Jakobs: Towards User-Friendly Networking
1988-11 * Kai Jakobs: The Directory - Evolution of a Standard
1988-12 * Kai Jakobs: Directory Services in Distributed Systems - A Survey
1988-13 * Martine Schümmer: RS-511, a Protocol for the Plant Floor
1988-14 * U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing
1988-15 * Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation
1988-16 * Fachgruppe Informatik: Jahresbericht 1987
1988-17 * Wolfgang Thomas: Automata on Infinite Objects
1988-18 * Michael Sonnenschein: On Petri Nets and Data Flow Graphs
1988-19 * Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers
1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen
1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment
1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers
1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies
1988-24 * Uschi Heuter: Generalized Definite Tree Languages
1989-01 * Fachgruppe Informatik: Jahresbericht 1988
1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik
1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions
1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language
1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)
1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?
1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems
1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control
1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial
1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation
1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science
1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?
1989-13 * Martine Schümmer: CNC/DNC Communication with MAP
1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements

1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments

1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling

1989-18 A. Maassen: Programming with Higher Order Functions

1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL

1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language

1990-01 * Fachgruppe Informatik: Jahresbericht 1989

1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)

1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas

1990-04 R. Loogen: Stack-based Implementation of Narrowing

1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies

1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication

1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work

1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation

1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke

1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine

1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit

1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains

1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)

1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars

1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars

1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit

1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen

1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations

1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences

1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming

1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks
1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks
1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30 T. Margaria: First-Order theories for the verification of complex FSMs
1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991

1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen

1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability

1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes

1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems

1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line

1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme

1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management

1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines

1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking

1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD

1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars

1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)

1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik

1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual

1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems

1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages

1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)

1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine

1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus

1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions

1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code

1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine

1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)

1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Red^+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus

1992-19-09 D. Howe, G. Burn: Experiments with strict STG code
1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes
1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine
1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction
1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)
1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer
1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine
1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell
1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation
1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages
1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)
1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment
1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)
1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers
1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)
1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)
1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting
1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)
1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models
1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures
1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)
1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language
1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language
1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing
1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free
1992-24 K. Pohl: The Three Dimensions of Requirements Engineering
1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications
1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification

1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety

1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design

1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic

1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems

1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications

1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice

1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN

1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis

1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models

1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES

1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme

1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks

1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems

1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components

1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages

1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications

1993-01 * Fachgruppe Informatik: Jahresbericht 1992

1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems

1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments

1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES

1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis

1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik

1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions

1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases

1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking

1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels

1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages

1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager

1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept

1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain

1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes

1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing

1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel

1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control

1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle

1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993

1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications

1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse

1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations

1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics

1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations

1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository

1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments

1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems

1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems

1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars

1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems

1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition

1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents
1994-16 P. Klein: Designing Software with Modula-3
1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words
1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction
1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas
1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)
1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras
1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry
1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach
1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach
1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment
1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments
1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes
1995-01 * Fachgruppe Informatik: Jahresbericht 1994
1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES
1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction
1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries
1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types
1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases
1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies
1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures
1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages
1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency
1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases
1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views

1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems

1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming

1996-01 * Jahresbericht 1995

1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees

1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates

1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability

1996-05 Klaus Pohl: Requirements Engineering: An Overview

1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools

1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs

1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics

1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming

1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents

1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines

1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation

1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell

1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems

1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming

1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management

1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement

1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models

1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools

1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases

1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation

1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations

1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems

1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto

1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design

1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996

1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization

1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting

1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets

1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations

1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

1998-01 * Fachgruppe Informatik: Jahresbericht 1997

1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes

1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments

1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future

1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases

1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML

1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

1999-01 * Jahresbericht 1998

1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager

1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference

1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie

1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL

1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge, Marcin Jurdzinski A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-03 D. Jäger, A. Schleicher, B. Westfechtel: UPGRADE: A Framework for Building Graph-Based Software Engineering Tools

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages
2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12	Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning
2006-13	Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-14	Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"
2006-15	Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16	Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17	Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01 *	Fachgruppe Informatik: Jahresbericht 2006
2007-02	Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations
2007-03	Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase
2007-04	Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation
2007-05	Uwe Naumann: On Optimal DAG Reversal
2007-06	Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking
2007-07	Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications
2007-08	Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches
2007-09	Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption
2007-10	Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes
2007-11	Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke
2007-12	Uwe Naumann: An L-Attributed Grammar for Adjoint Code
2007-13	Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes
2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs
2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks
2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting
2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete
2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control
2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems
2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains
2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets
2008-01 * Fachgruppe Informatik: Jahresbericht 2007/2008
2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing
2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination
2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler
2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations
2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs
2008-07 Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition
2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs
2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs
2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time
2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows
2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages
2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs
2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers
2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2008-19 Dirk Wilking: Empirical Studies for the Application of Agile Methods to Embedded Systems

2009-01 * Fachgruppe Informatik: Jahresbericht 2009

2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications

2009-03 Alexander Nyßen: Model-Based Construction of Embedded & Real-Time Software - A Methodology for Small Devices

2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs

2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete

2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I

2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs

2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem

2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm

2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs

2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes

2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games

2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)

2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems

2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs

2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata

2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies

2010-01 * Fachgruppe Informatik: Jahresbericht 2010

2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time

2010-03 Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering

2010-04	René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme
2010-05	Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme
2010-06	Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata
2010-07	George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms
2010-08	Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting
2010-09	George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs
2010-10	Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut
2010-11	Martin Zimmermann: Parametric LTL Games
2010-12	Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut
2010-13	Ashraf Armoush: Design Patterns for Safety-Critical Embedded Systems
2010-14	Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination
2010-15	Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode
2010-16	Christian Berger: Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles
2010-17	Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten
2010-18	Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit
2010-19	Felix Reidl: Experimental Evaluation of an Independent Set Algorithm
2010-20	Wladimir Fridman, Christof Löding, Martin Zimmermann: Degrees of Lookahead in Context-free Infinite Games
2011-01 *	Fachgruppe Informatik: Jahresbericht 2011
2011-02	Marc Brockschmidt, Carsten Otto, Jürgen Giesl: Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting
2011-03	Lars Noschinski, Fabian Emmes, Jürgen Giesl: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
2011-04	Christina Jansen, Jonathan Heinen, Joost-Pieter Katoen, Thomas Noll: A Local Greibach Normal Form for Hyperedge Replacement Grammars
2011-06	Johannes Lotz, Klaus Leppkes, and Uwe Naumann: dco/c++ - Derivative Code by Overloading in C++
2011-07	Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe: An Operational Semantics for Activity Diagrams using SMV
2011-08	Thomas Ströder, Fabian Emmes, Peter Schneider-Kamp, Jürgen Giesl, Carsten Fuhs: A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog
2011-09	Markus Beckers, Johannes Lotz, Viktor Mosenkis, Uwe Naumann (Editors): Fifth SIAM Workshop on Combinatorial Scientific Computing

2011-10	Markus Beckers, Viktor Mosenkis, Michael Maier, Uwe Naumann: Adjoint Subgradient Calculation for McCormick Relaxations
2011-11	Nils Jansen, Erika Ábrahám, Jens Katelaan, Ralf Wimmer, Joost-Pieter Katoen, Bernd Becker: Hierarchical Counterexamples for Discrete-Time Markov Chains
2011-12	Ingo Felscher, Wolfgang Thomas: On Compositional Failure Detection in Structured Transition Systems
2011-13	Michael Förster, Uwe Naumann, Jean Utke: Toward Adjoint OpenMP
2011-14	Daniel Neider, Roman Rabinovich, Martin Zimmermann: Solving Muller Games via Safety Games
2011-16	Niloofar Safiran, Uwe Naumann: Toward Adjoint OpenFOAM
2011-17	Carsten Fuhs: SAT Encodings: From Constraint-Based Termination Analysis to Circuit Synthesis
2011-18	Kamal Barakat: Introducing Timers to pi-Calculus
2011-19	Marc Brockschmidt, Thomas Ströder, Carsten Otto, Jürgen Giesl: Automated Detection of Non-Termination and NullPointerExceptions for Java Bytecode
2011-24	Callum Corbett, Uwe Naumann, Alexander Mitsos: Demonstration of a Branch-and-Bound Algorithm for Global Optimization using McCormick Relaxations
2011-25	Callum Corbett, Michael Maier, Markus Beckers, Uwe Naumann, Amin Ghobeity, Alexander Mitsos: Compiler-Generated Subgradient Code for McCormick Relaxations
2011-26	Hongfei Fu: The Complexity of Deciding a Behavioural Pseudometric on Probabilistic Automata
2012-01	Fachgruppe Informatik: Annual Report 2012
2012-02	Thomas Heer: Controlling Development Processes
2012-03	Arne Haber, Jan Oliver Ringert, Bernhard Rumpe: MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems
2012-04	Marcus Gelderie: Strategy Machines and their Complexity
2012-05	Thomas Ströder, Fabian Emmes, Jürgen Giesl, Peter Schneider-Kamp, and Carsten Fuhs: Automated Complexity Analysis for Prolog by Term Rewriting
2012-06	Marc Brockschmidt, Richard Musiol, Carsten Otto, Jürgen Giesl: Automated Termination Proofs for Java Programs with Cyclic Data
2012-07	André Egners, Björn Marschollek, and Ulrike Meyer: Hackers in Your Pocket: A Survey of Smartphone Security Across Platforms
2012-08	Hongfei Fu: Computing Game Metrics on Markov Decision Processes
2012-09	Dennis Guck, Tingting Han, Joost-Pieter Katoen, and Martin R. Neuhäußer: Quantitative Timed Analysis of Interactive Markov Chains
2012-10	Uwe Naumann and Johannes Lotz: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Direct Solvers for Systems of Linear Equations
2012-12	Jürgen Giesl, Thomas Ströder, Peter Schneider-Kamp, Fabian Emmes, and Carsten Fuhs: Symbolic Evaluation Graphs and Term Rewriting — A General Methodology for Analyzing Logic Programs

2012-15 Uwe Naumann, Johannes Lotz, Klaus Leppkes, and Markus Towara: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Solvers for Systems of Nonlinear Equations

2012-16 Georg Neugebauer and Ulrike Meyer: SMC-MuSe: A Framework for Secure Multi-Party Computation on MultiSets

2012-17 Viet Yen Nguyen: Trustworthy Spacecraft Design Using Formal Methods

2013-01 * Fachgruppe Informatik: Annual Report 2013

2013-02 Michael Reke: Modellbasierte Entwicklung automobiler Steuerungssysteme in Klein- und mittelständischen Unternehmen

2013-03 Markus Towara and Uwe Naumann: A Discrete Adjoint Model for OpenFOAM

2013-04 Max Sagebaum, Nicolas R. Gauger, Uwe Naumann, Johannes Lotz, and Klaus Leppkes: Algorithmic Differentiation of a Complex C++ Code with Underlying Libraries

2013-05 Andreas Rausch and Marc Sihling: Software & Systems Engineering Essentials 2013

2013-06 Marc Brockschmidt, Byron Cook, and Carsten Fuhs: Better termination proving through cooperation

2013-07 André Stollenwerk: Ein modellbasiertes Sicherheitskonzept für die extrakorporale Lungenunterstützung

2013-08 Sebastian Junges, Ulrich Loup, Florian Corzilius and Erika Ábrahám: On Gröbner Bases in the Context of Satisfiability-Modulo-Theories Solving over the Real Numbers

2013-10 Joost-Pieter Katoen, Thomas Noll, Thomas Santen, Dirk Seifert, and Hao Wu: Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata

2013-12 Marc Brockschmidt, Fabian Emmes, Stephan Falke, Carsten Fuhs, and Jürgen Giesl: Alternating Runtime and Size Complexity Analysis of Integer Programs

2013-13 Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle: SensorCloud: Towards the Interdisciplinary Development of a Trustworthy Platform for Globally Interconnected Sensors and Actuators

2013-14 Jörg Brauer: Automatic Abstraction for Bit-Vectors using Decision Procedures

2013-16 Carsten Otto: Java Program Analysis by Symbolic Execution

2013-19 Florian Schmidt, David Orlea, and Klaus Wehrle: Support for error tolerance in the Real-Time Transport Protocol

2013-20 Jacob Palczynski: Time-Continuous Behaviour Comparison Based on Abstract Models

2014-01 * Fachgruppe Informatik: Annual Report 2014

2014-02 Daniel Merschen: Integration und Analyse von Artefakten in der modellbasierten Entwicklung eingebetteter Software

2014-03 Uwe Naumann, Klaus Leppkes, and Johannes Lotz: dco/c++ User Guide

2014-04 Namit Chaturvedi: Languages of Infinite Traces and Deterministic Asynchronous Automata

2014-05 Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, and Peter Schneider-Kamp: Automated Termination Analysis for Programs with Pointer Arithmetic

2014-06 Esther Horbert, Germán Martín García, Simone Frintrop, and Bastian Leibe: Sequence Level Salient Object Proposals for Generic Object Detection in Video

2014-07 Niloofar Safiran, Johannes Lotz, and Uwe Naumann: Algorithmic Differentiation of Numerical Methods: Second-Order Tangent and Adjoint Solvers for Systems of Parametrized Nonlinear Equations

2014-08 Christina Jansen, Florian Göbe, and Thomas Noll: Generating Inductive Predicates for Symbolic Execution of Pointer-Manipulating Programs

2014-09 Thomas Ströder and Terrance Swift (Editors): Proceedings of the International Joint Workshop on Implementation of Constraint and Logic Programming Systems and Logic-based Methods in Programming Environments 2014

2014-14 Florian Schmidt, Matteo Ceriotti, Niklas Hauser, and Klaus Wehrle: HotBox: Testing Temperature Effects in Sensor Networks

2014-15 Dominique Gückel: Synthesis of State Space Generators for Model Checking Microcontroller Code

2014-16 Hongfei Fu: Verifying Probabilistic Systems: New Algorithms and Complexity Results

2015-01 * Fachgruppe Informatik: Annual Report 2015

2015-02 Dominik Franke: Testing Life Cycle-related Properties of Mobile Applications

2015-05 Florian Frohn, Jürgen Giesl, Jera Hensel, Cornelius Aschermann, and Thomas Ströder: Inferring Lower Bounds for Runtime Complexity

2015-06 Thomas Ströder and Wolfgang Thomas (Editors): Proceedings of the Young Researchers' Conference "Frontiers of Formal Methods"

2015-07 Hilal Diab: Experimental Validation and Mathematical Analysis of Cooperative Vehicles in a Platoon

2015-08 Mathias Pelka, Jó Agila Bitsch, Horst Hellbrück, and Klaus Wehrle (Editors): Proceedings of the 1st KuVS Expert Talk on Localization

2015-09 Xin Chen: Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models

2015-11 Stefan Wüller, Marián Kühnel, and Ulrike Meyer: Information Hiding in the Public RSA Modulus

2015-12 Christoph Matheja, Christina Jansen, and Thomas Noll: Tree-like Grammars and Separation Logic

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.