Preprint of the paper which appeard in: M. Breuer, F. Durst, Chr. Zenger (eds.): High Performance Scientific and Engineering Computing, Lecture Notes in Computational Science and Engineering 21 (Springer-Verlag, 2002) 333-340.

Modeling Techniques and Parameter Estimation for the Simulation of Complex Vehicle Structures

Torsten Butz[1,2], Oskar von Stryk[2], Cornelius Chucholowski[1], Stephan Truskawa[1], and Thieß-Magnus Wolter[1]

[1] TESIS *DYNAware* GmbH, Implerstraße 26, D-81371 München, www.tesis.de
[2] FG Simulation und Systemoptimierung, Technische Universität Darmstadt, Alexanderstr. 10, D-64283 Darmstadt, www.sim.informatik.tu-darmstadt.de

Abstract. The numerical simulation of complex vehicle structures requires dynamic models for passenger cars as well as for trucks and vehicles with trailers. Tailored numerical modeling and integration techniques must be employed to achieve real-time capability of the considered vehicle dynamics program which is vital for its use within hardware-in-the-loop test-benches. To efficiently calibrate the vehicle model a parameter estimation tool was developed which relies on observations obtained from driving tests. Combining robust nonlinear optimization algorithms and careful numerical differentiation it is well suited for low-cost parallel computing platforms, such as heterogeneous PC clusters, which are usually available for automotive suppliers and industries employing vehicle dynamics simulations.

1 Introduction

Simulations of the full vehicle dynamics play a key role in automotive development, since they enable the road performance and the handling properties of a new car to be investigated in advance. Besides reducing the need for physical prototyping, numerical simulations may be used within software- and hardware-in-the-loop test-benches, which allow control units, such as antilock braking systems and electronic stability programs, to be tested without danger for test driver and vehicle.

The development of intricate control devices and strategies requires the virtual car to reproduce the complex behavior of the real vehicle in detail. Therefore, we employ a sophisticated vehicle model which comprises a suitable multi-body system as well as a realistic tire model. In this paper, the model is extended to trucks and vehicles with trailers. The use of tailored modeling and numerical integration techniques enables the entire vehicle dynamics to be described by a large system of ordinary differential equations which can be solved in real time.

The calibration of the model substantially relies on the application of robust nonlinear optimization algorithms and careful numerical differentiation. The resulting parameter estimation scheme allows model coefficients

which are not available from physical measurements to be determined efficiently. The developed program tool is suitable for the parallel use on low-cost computing platforms such as heterogeneous PC networks. It was successfully applied to adjust selected vehicle model parameters for a passenger car.

2 Simulation of Full Vehicle Dynamics

The vehicle dynamics program *veDYNA* [10] which forms the basis of the following investigations is developed and commercially distributed by TESIS *DYNAware*, München. The integration of the program core into a Matlab/Simulink [12,13] environment enables versatile application and easy handling through a graphical user interface.

The vehicle model in *veDYNA* consists of a system of rigid bodies which comprise the vehicle body, the axle suspensions and the wheels. Additional partial models depict the characteristics of the drive train, the steering mechanism and the tires [9]. While general purpose methods for modeling multibody dynamics would yield a differential-algebraic system of index 3, we make use of appropriate minimum coordinates which avoid algebraic constraints in the equations of motion

$$M_{BV}\,\dot{z}_{BV} = Q_{BV}(y_{BV}, z_{BV}, y_{ST}, z_{ST}, y_{DT}, z_{DT}) \tag{1}$$
$$\dot{y}_{BV} = K_{BV}^{-1}(y_{BV})\; z_{BV} \tag{2}$$
$$M_{DT}\,\dot{z}_{DT} = Q_{DT}(y_{DT}, z_{DT}) \tag{3}$$
$$\dot{y}_{DT} = V_{DT}\, z_{DT} \tag{4}$$
$$M_{ST}\,\dot{z}_{ST} = Q_{ST}(y_{ST}, z_{ST}) \tag{5}$$
$$\dot{y}_{ST} = V_{ST}\, z_{ST} \tag{6}$$
$$D\,\dot{y}_T = F_{stat} \;-\; C\, y_T. \tag{7}$$

Thus, the vehicle dynamics in *veDYNA* is fully described by a system of first-order differential equations comprising 24 ODEs (1), (2) for the basic vehicle, 19 equations (3), (4) governing the dynamics of the drive train, five ODEs (5), (6) for the steering system as well as eight additional ODEs (7) which depict the deviations of the tires. Couplings between the separate systems occur by way of the generalized forces and torques Q_{BV}.

The basic vehicle model is now extended to vehicles with two rear axles and vehicles with trailers. Thus, it is possible to analyze the dynamics of single trucks as well as of trucks and passenger cars with trailers. Besides investigating stability issues in truck and trailer design, this feature allows the development of anti-roll control strategies for semi-trailers. Moreover, *veDYNA* may be employed in a hardware-in-the-loop setup to investigate anti-lock braking systems for vehicles with caravans (cf. Fig. 1).

In *veDYNA* the trailer is treated as a separate vehicle which lacks a drive train and is coupled mechanically to the leading vehicle. The transmission of

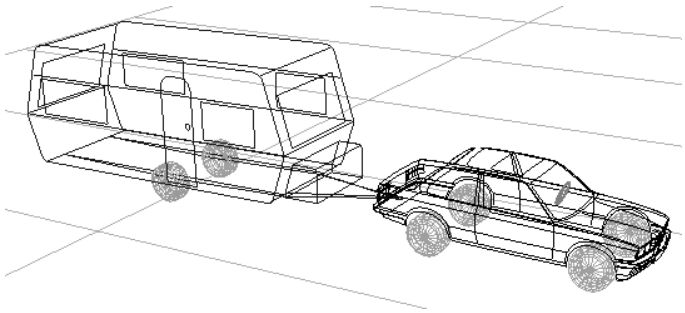

Fig. 1. *veDYNA* model for vehicles with trailers

the driving moment to the trailer is modeled by elastic and frictional forces whose strengths depend on the geometry and the rigidity of the coupling. To compute the magnitude of these forces a set of local coordinates and angles is introduced which allow the relative position and orientation of vehicle and trailer to be determined.

For a realistic implementation of virtual driving tests on the computer also a nonlinear driver model and a program module for the simulation of arbitrary road geometries and conditions were developed [3,10].

The equations of motion for vehicle, trailer and driver result in a system of stiff ordinary differential equations which need to be integrated with a specifically tailored algorithm. The use of a semi-implicit Euler scheme [9] enables a stable numerical solution for integration steps of several milliseconds. Due to the absence of drive train and driver in the model for the trailer the computational complexity of the coupled vehicle models is reduced proportionately. Thus, step sizes in the range of few milliseconds allow real-time simulations of complex vehicle structures on current PC hardware.

3 Estimation of Vehicle Model Parameters

The system of differential equations describing the dynamics of vehicle, trailer and driver in *veDYNA* may be summarized by

$$\dot{x}(t) = g(x(t), u(t), p, t) \tag{8}$$

with suitable initial values $x(t_0) = x_0$. In addition to the vehicle's state variables $x(t) \in \mathbb{R}^{n_x}$ comprising the generalized coordinates and velocities, the full-vehicle performance is also governed by a number of control variables $u(t) \in \mathbb{R}^{n_u}$ which determine the respective driving maneuver. The unknown model parameters $p \in \mathbb{R}^{n_p}$ are constant for all times t.

The calibration of the *veDYNA* vehicle model aims to determine model coefficients of the virtual car which are not directly available from physical measurements. Suitable parameter values shall be estimated such that the

numerical integration results closely fit the observed behavior of a physical prototype. Consequently, we obtain the nonlinear least-squares problem

$$\underset{p\in\mathbb{R}^{n_p}}{\text{minimize}}\; r(p) := \frac{1}{2}\,\|f(p)\|_2^2 := \frac{1}{2}\sum_{j=1}^{n_t}\sum_{i\in I_j}\left(\eta_{ij} - x_i(t_j,p)\right)^2 \tag{9}$$

where the η_{ij}, $i \in I_j$, are measurements of selected vehicle state variables recorded at the times t_j during a driving test, and $x(t,p)$ denotes the corresponding numerical solution for a specific parameter set $p \in \mathbb{R}^{n_p}$. Usually, additional box constraints

$$l_i \le p_i \le u_i, \quad i = 1, ..., n_p, \tag{10}$$

on the parameter range have to be considered which shall ensure optimization results compatible with the real vehicle properties.

For the solution of (9), (10) a parameter estimation tool was developed which allows the application of robust mathematical optimization algorithms [1,2]. In addition, a Matlab implementation is now provided which enables comfortable pre-processing of the experimental data and easy handling of the numerical optimization codes through a graphical user interface. Moreover, it supports the visualization of the identification results as well as a mathematical sensitivity analysis, which can be used to determine the impact of single parameters on (9) and the reliability of their numerical estimates.

Currently, the employed optimization algorithms consist of the implementation LMDER of the Levenberg-Marquardt method [7], which is based on the iterative solution of approximate linear least-squares problems, and the implicit filtering code IFFCO [6], a quasi-Newton algorithm for noisy optimization problems. Both algorithms are available from the World Wide Web and were furnished with an application program interface for Matlab which allows them to be called from the Matlab prompt.

Moreover, the general design of the user interface also enables a variety of other methods, as for instance from the Matlab Optimization Toolbox [11], to be incorporated easily. As an example, two gradient-free search techniques were employed for the parameter estimation. The Matlab implementation FMINS of the Nelder-Mead simplex algorithm was extended to handle parallel objective evaluations and problems with simple bounds (10). Further, the evolutionary strategy EVOCLASS due to [8] was implemented whereupon a parallel interface and the treatment of box constraints were introduced.

For the optimization with gradient-based methods the Jacobian matrix $\partial f/\partial p$ of the objective (9) must be supplied. However, the complexity of the underlying vehicle model does not support analytic or internal numerical differentiation techniques for this purpose. Rather the required first-order derivatives need to be determined by means of finite difference approximations where the increments must be chosen carefully such as to account for truncation, condition, and rounding errors [5].

For LMDER the partial derivatives $\partial_i f(p)$ with respect to the i-th parameter are obtained from the one-sided differences

$$(\partial_i f(p))_{\pm h} = \frac{f(p \pm he_i) - f(p)}{\pm h}, \tag{11}$$

depending on the feasibility of $p + he_i$ or $p - he_i$. Further, the variable finite difference strategy in IFFCO is based on central differences

$$(\partial_i r(p))_{2h} = \frac{r(p + he_i) - r(p - he_i)}{2h} \tag{12}$$

provided that both $p + he_i$ and $p - he_i$ are feasible; otherwise (11) is used as well. In both cases, $e_i \in \mathbb{R}^{n_p}$ denotes the i-th canonical unit vector, and $h > 0$ is a suitable finite difference increment.

Obviously, this strategy entails a large number of additional evaluations of the objective function depending on the derivative approximation being used. But also the application of gradient-free methods often requires multiple function evaluations to be performed at the same time, when for instance the Nelder-Mead simplex or new populations for the evolutionary strategy are generated. Accordingly, the computational time for the optimization is significantly reduced by distributing the simultaneous objective evaluations among several processors.

As to the parallel platform a heterogeneous PC cluster with MS Windows operating systems was presumed which is well-priced and usually available for small and medium-sized automotive suppliers and industries. The communication of data across the network was realized by means of the ONC RPC library from Sun Microsystems, ported to MS Windows systems [4]. Matlab application program interfaces for the relevant routines were developed making the parallel setup available from the Matlab prompt. Thus, the parallel distribution of simultaneous function evaluations and the management of the RPC network is enabled through a graphical user interface for Matlab.

4 Results

In [1,2] the numerical parameter estimation scheme was successfully employed to determine selected coefficients for the vehicle model of a passenger car; though, the parameter estimation had been carried out with a stand-alone console application of *veDYNA* and the associated optimization frame. Here, we present results which were obtained from the newly developed parameter estimation tool for the Matlab/Simulink emulation of *veDYNA*.

As an example, we consider the calibration of the brake friction coefficients of the vehicle model which indicate the ratio for the generated braking moments transmitted to the wheels. Since the brakes in *veDYNA* are uniformly modeled as disc brakes, the adjustment of these parameters to the actual brake mechanism represents a common problem in practice. The remaining

parameters of the *veDYNA* vehicle model as well as the underlying maneuver data were provided by an automotive manufacturer for testing purposes.

The recorded measurements consisted of the brake pressures and the rotational wheel speeds throughout the full braking of a passenger car equipped with an anti-lock braking system. The actual braking was preceded by a speed-up phase where the vehicle was accelerated to the experimentally observed initial speed of approximately 106.8 km/h. Thus, the replication of the driving maneuver by simulation resulted in a total maneuver time of 20.25 s where 17.1 s and 3.15 s were required for the acceleration and the braking phase respectively. For the purpose of a realistic simulation the observed brake pressures which were measured every 0.001 s were used as input control variables. The nonlinear least-squares criterion (9) consisted of the experimental wheel speeds, also recorded at regular intervals of 0.001 s, set off against the corresponding numerical integration results, which yields a total number of 12600 addends.

Starting with initial values

$$p^0 = (0.45, 0.45, 0.45, 0.45)^T \tag{13}$$

which were given by the default values from the provided vehicle database, our parameter identification tool was used to determine suitable estimates for the four unknown coefficients. Here, the initial guesses (13) refer to the friction coefficients at the left and right front wheels and the left and right rear wheels respectively. The associated least-squares residual (9) was given by $r(p^0) = 6.374 \cdot 10^6$.

Due to the optimization the objective value was reduced to $r(p^*) = 7.713 \cdot 10^4$ which is approximately one percent of its initial size. The minimum residual was assumed for the parameter values

$$p^* = (0.2505, 0.2670, 0.4253, 0.3662)^T \ . \tag{14}$$

A numerical sensitivity analysis revealed small confidence intervals indicating that reliable estimates were obtained. The deviations between the coefficients (14) at the rear wheels allow for the different wear of the respective braking devices and tires as well as for possible measurement errors.

A comparison between the measured wheel speeds and the corresponding simulation results for the optimal solution (14) is depicted in Fig. 2. Obviously, good agreement was achieved for the respective characteristics. The remaining deviations between the computed and the experimentally observed values must be attributed to the uneven surface of the test track which is not depicted in detail by the road model of *veDYNA*.

The quoted computations were carried out on a homogeneous PC cluster consisting of eight Intel Pentium 500 MHz processors with Windows NT 4.0 operating systems. Each evaluation of the least-squares residual (9), which comprised a full vehicle dynamics simulation with *veDYNA* and a record of the simulation data, required a computational time of about 45 seconds. The

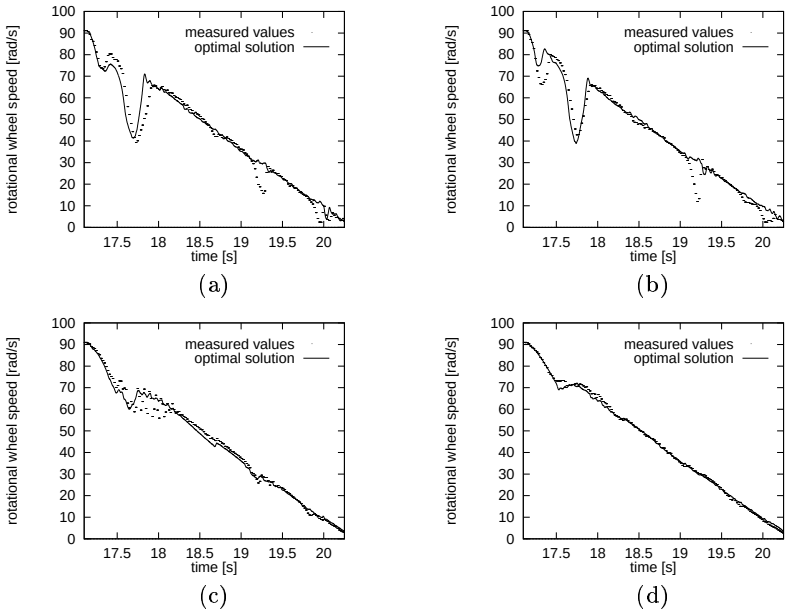

Fig. 2. Comparison between observed and computed rotational speeds for the left (**a**) and right (**b**) front wheels and the left (**c**) and right (**d**) rear wheels

software packages Matlab 5.3 and Simulink Toolbox 3.0 as well as the present version *veDYNA* 3.3.2 were used.

For the numerical solution of the parameter estimation problem we employed the Matlab ports of LMDER und IFFCO as well as the extended simplex algorithm FMINSB and the newly developed evolutionary strategy EVOCLASS. The latter pursued a (3,8)-strategy where three current parameter sets served to generate eight new estimates in each iteration.

For comparison, the numerical optimization was carried out sequentially as well as in parallel. The parallel optimization with LMDER and FMINSB required four processors treating the additional function evaluations for the one-sided differences (11) and the geometrical transformations of the simplex. For the application of IFFCO and EVOCLASS eight processors were supplied which served to compute the symmetric differences (12) and the respective new generations. Dynamic load sharing was implemented such as to achieve minimum CPU times.

Table 1 compares the different optimization codes, giving the least-squares residual $r(p^*)$ for the respective solutions, and the computational times t_{seq} and t_{par} for the sequential and parallel executions of the parameter estimation. The ratios of these CPU times, i. e., the achieved parallel speed-ups, are shown in the last column. Also listed are the numbers of objective evaluations n_{seq} and n_{par} performed during the entire optimization and by the client process in the parallel framework respectively.

Table 1. Computational results for the employed optimization algorithms

Optimization code	$r(p^*)$	n_{seq}	t_{seq} [s]	n_{par}	t_{par} [s]	$n_{\text{seq}}/n_{\text{par}}$	$t_{\text{seq}}/t_{\text{par}}$
LMDER	$7.730 \cdot 10^4$	43	1946.1	19	897.0	2.26	2.17
IFFCO	$7.726 \cdot 10^4$	288	12070.0	103	4410.3	2.80	2.74
FMINSB	$7.733 \cdot 10^4$	234	9987.9	134	5961.8	1.75	1.68
EVOCLASS	$7.716 \cdot 10^4$	1201	52225.8	151	6715.6	7.95	7.78

For this problem, all investigated algorithms produced reasonably small residuals. The parallel treatment of the finite difference computations reduced the CPU time for both gradient-based codes by more than a factor of two. The maximum parallel speed-up is achieved by the evolutionary strategy where each new generation requires a corresponding number of simultaneous evaluations of the objective function. Accordingly, the parallel approach achieves competitive computational times also for the gradient-free methods.

References

1. T. Butz, O. von Stryk, M. Vögel, T.-M. Wolter, C. Chucholowski: *Parallel parameter estimation in full motor vehicle dynamics.* SIAM News 33, 4 (2000)
2. T. Butz, O. von Stryk, T.-M. Wolter: *A parallel optimization scheme for parameter estimation in motor vehicle dynamics.* In: A. Bode et al. (eds.): Euro-Par 2000 - Parallel Processing. Lecture Notes in Computer Science 1900, Springer, Berlin (2000) 829–834
3. C. Chucholowski, M. Vögel, O. von Stryk, T.-M. Wolter: *Real time simulation and online control for virtual test drives of cars.* In: H.-J. Bungartz et al. (eds.): High Performance Scientific and Engineering Computing. Lecture Notes in Computational Science and Engineering 8, Springer, Berlin (1999) 157–166
4. M. Gergeleit: *ONC RPC for Windows NT Homepage.* World Wide Web, http://www.dcs.qmw.ac.uk/~williams/nisgina-current/src/rpc110/oncrpc.htm (1996)
5. P. E. Gill, W. Murray, M. H. Wright: *Practical Optimization.* Academic Press, London New York (1981)
6. P. Gilmore: *IFFCO: Implicit Filtering for Constrained Optimization, User's Guide.* Technical Report CRSC-TR93-7, Center for Research in Scientific Computation, North Carolina State University, Raleigh (1993)
7. J. J. Moré: *The Levenberg-Marquardt Algorithm: Implementation and Theory.* In: A. Dold, B. Eckmann (eds.): Numerical Analysis. Lecture Notes in Mathematics 630. Springer, Berlin Heidelberg (1978) 105–116
8. I. Rechenberg: Evolutionsstrategie. Frommann-Holzboog, Stuttgart (1994)
9. G. Rill: *Simulation von Kraftfahrzeugen.* Vieweg, Braunschweig (1994)
10. TESIS *DYNAware*: *veDYNA User's Guide.* München (1997)
11. The MathWorks Inc.: *Optimization Toolbox User's Guide.* Nattick (1999)
12. The MathWorks Inc.: *Using MATLAB.* Nattick (1999)
13. The MathWorks Inc.: *Using Simulink.* Nattick (1999)