Mathematical Modelling of Systems
1997, Vol. 3, No. 2, pp. 000–111

1381-2424/97/0301-00$12.00

Approximation of contact geometry in the dynamical simulation of wheel-rail systems

M. ARNOLD* AND H. NETTER*

ABSTRACT

Models for the geometrical contact of wheel and rail are a basic component of multi-body system (MBS) models for wheel-rail systems. Approximations are used to get sufficiently differentiable contact conditions that can be evaluated efficiently. We discuss an approximation that is essentially based on polynomial 2D-tensorproduct splines and minimizes a functional that combines a weighted least squares approximation of a rigid contact model with a smoothing term. An efficient algorithm to compute the tensorproduct spline is developed, the parallelization on a cluster of workstations is discussed. We report on results of both the sequential and the parallel algorithm and give simulation results for a rigid wheelset on a straight track.

Key words: dynamical simulation, multibody systems, PVM, tensorproduct spline, wheel-rail contact

Subject classification (AMS): 65C20, 65D07, 65Y05

1 INTRODUCTION

Dynamical simulation is a more and more essential part in the design and analysis process of advanced railway vehicles. Within simulation packages the dynamical behaviour of wheel-rail systems has to be analysed for large sets of parameters. This motivates the development of robust tools for the efficient dynamical simulation of wheel-rail systems. In this field the mechanical multi-body system (MBS) approach is well established: The motion of the wheel-rail system is described by the Euler–Lagrange equations

$$\begin{aligned} M(q)\ddot{q} &= f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda \\ 0 &= g(q,t) \end{aligned} \tag{1}$$

*German Aerospace Research Establishment, Institute of Robotics and System Dynamics, DLR Oberpfaffenhofen, P.O. Box 1116, D – 82230 Wessling, Germany,

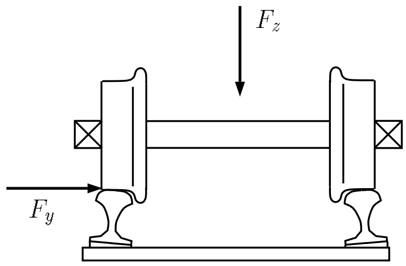

Fig. 1. Rigid wheelset on a straight track.

in the position coordinates q. Here $M(q)$ denotes the symmetric, positive definite mass matrix and $f(q,\dot{q},\lambda,t)$ the applied forces and friction forces, which depend on the constraint forces $-G^T(q,t)\lambda$. The constraints $g(q,t)=0$ are coupled to the dynamical equations by $G(q,t) := g_q(q,t)$ and the Lagrangian multipliers λ. They guarantee that the wheels are always in contact with the rails (*contact conditions*). The dynamical equations in (1) result from the conditions that guarantee the equilibrium of forces and momenta in the wheel-rail system. Forces, that have to be considered in typical applications, include e. g., gravitational forces and coupling forces in vehicle suspensions.

For some model problems (rigid wheelsets, bogies) the equations of motion (1) have been given in detail in the literature (e. g., [4], [19]). For more complex wheel-rail systems the model equations are generated using multibody formalisms (e. g., [17]).

In Section 5 we consider as an example the motion of the rigid wheelset that is shown in Fig. 1. The forces F_z and F_y are introduced to model the axle load (F_z) and an additional force (F_y) that acts in lateral direction (see Example 4). The actual position of the wheelset in an inertial system is described by $q(t) \in \mathbb{R}^6$ (corresponding to the 6 degrees of freedom). The motion is restricted by 2 contact conditions $g_1(q,t) = g_2(q,t) = 0$ such that neither the left nor the right wheel leaves the rails (see also Section 5).

Eqs. (1) form a differential-algebraic system of index 3. Theory, numerical analysis and software for (1) are well developed (see e. g., [3] and the references therein). We do not focus on this point in the present paper and note only that (1) contains inherently derivatives of the contact condition $g(q,t)=0$. The Lagrangian multipliers λ are determined by the second derivatives of the constraints:

$$\begin{aligned} 0 &= \frac{d^2}{dt^2} g(q,t) = \frac{d}{dt}(G(q,t)\dot{q} + g_t)\ , \\ &= [GM^{-1}](q)(f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda) + g_{qq}(\dot{q},\dot{q}) + g_{qt}\dot{q} + g_{tt}\ . \end{aligned} \tag{2}$$

To know: in general the contact conditions $g = 0$ have to be (at least) two times continuously differentiable to get continuous constraint forces $-G^T\lambda$.

The traditional description of wheel-rail contact in a MBS model is based on the geometry of two rigid bodies (wheel and rail) that get in contact in one isolated point that is called *contact point*. But in real life wheel and rail are elastically deformed in a neighbourhood of this "contact point" and there is a *contact patch* between wheel and rail. Size and shape of this contact patch have a great influence on the friction forces, i. e. on $f(q, \dot{q}, \lambda, t)$ (see [11] for a comprehensive study of this subject).

However, in the geometrical contact condition $g = 0$ the elastic deformation (that depends on the axle load and is in the range $10 \ldots 100\,\mu$m) can be neglected. This *rigid body contact model* avoids high-frequency oscillations in the solution and saves up to 95% cpu-time compared with a pure elastic contact model. Rigid body contact models have been used successfully in simulations with conic wheels (e. g., [19]) but they cause problems if the wheels have a *wear profile*. For these wheels (that are commonly used by European railway companies) the rigid body contact model results in constraints $g(q, t) = 0$ that are either not differentiable or hard to evaluate (see Section 2).

In the dynamical simulation we therefore use a sufficiently differentiable approximation of $g(q, t)$ that can be evaluated efficiently. This approximation is implemented in the industrial simulation package SIMPACK, as part of SIMPACK it is used in the industrial design of railway vehicles ([17]). We carefully selected an approximation that is adapted to the contact problem between a single wheel and a rail, that is flexible w. r. t. future model changes and that can be computed and evaluated efficiently and very robustly (see Section 2).

The essential part of this approximation are polynomial 2D-tensorproduct splines minimizing a functional that combines a weighted least squares approximation of the rigid body contact model with a smoothing term. In Section 3 we discuss efficient and numerically stable algorithms for the computation and evaluation of the tensorproduct splines. Modern software environments allow a drastic acceleration of the computation process. In Section 4 we report on a parallel implementation on a cluster of workstations using PVM ([8]).

In the simulation package the tensorproduct splines are computed once in a pre-processing step, the spline coefficients are stored and allow the evaluation of the spline during the dynamical simulation. With 2 examples we illustrate in Section 5 the advantages of approximation in the dynamical simulation. Compared with the rigid body contact model the cpu-time for the simulation is reduced by a factor $5 \ldots 12$.

2 APPROXIMATION OF RIGID BODY MODELS FOR WHEEL-RAIL CONTACT

As basic unit of wheel-rail contact we use the contact of a single wheel and a rail. This allows the simulation both for wheel-rail systems with classical rigid wheelsets and for modern advanced railway vehicles without rigid coupling between the wheels of a wheelset. In this section we describe 2 rigid body contact models and motivate the use of tensorproduct spline approximations in the contact model.

Throughout the present paper we restrict ourselves to problems with rails that have non-varying profiles along the track. This includes motion on straight tracks and motion into or in curves but excludes e. g., the motion of a railway vehicle running through a switch. Under this restriction the rails can be approximated locally by a straight track. We therefore have to consider the contact between a body of revolution (wheel) and a prismatic body (rail). The undeformed surfaces of wheel and rail are defined by industrial standards, in local systems of coordinates $\boldsymbol{W}$ and $\boldsymbol{R}$ they are given by

$$\{ \, (\, F(s) \sin \tau \, , \, s \, , \, F(s) \cos \tau \,)^T \, : \, \tau \in [0, 2\pi), \, \underline{s} \le s \le \overline{s} \, \}$$

and

$$\{ \, (\, x \, , \, y \, , \, G(y) \,)^T \, : \, x \in \mathbb{R}, \, \underline{y} \le y \le \overline{y} \, \} \, , \tag{3}$$

respectively. The profile functions $F(s)$ and $G(y)$ depend on the wheel coordinate s and the rail coordinate y. In $\boldsymbol{W}$ the s-axis is parallel to the wheel axle. System $\boldsymbol{R}$ is defined such that the x-axis is parallel to the track, the z-axis is oriented downwards and the origin of $\boldsymbol{W}$ lies in the (y, z)-plane (see Fig. 2).

If we neglect for the moment the elastic deformation then the position of the wheel as rigid body is completely determined by the position $(x_{\mathrm{W}}, y_{\mathrm{W}}, z_{\mathrm{W}})^T$ of the origin of $\boldsymbol{W}$ in $\boldsymbol{R}$ ($x_{\mathrm{W}} = 0$ by construction) and by 3 angles of rotation φ, ϑ, ψ that represent rotations around the x-, y-, and z-axis, respectively. φ and ψ are defined such that the s-axis of $\boldsymbol{W}$ and the y-axis of $\boldsymbol{R}$ are parallel if $\varphi = \psi = 0$ (see figure).

In $\boldsymbol{R}$ the wheel surface is given by

$$\{ \, (x, y, z)^T := (0, y_{\mathrm{W}}, z_{\mathrm{W}})^T + A(\varphi, \psi) \cdot (F(s) \sin \tau, s, F(s) \cos \tau)^T \, \} \tag{4}$$

with a matrix $A(\varphi, \psi)$ that represents the rotations around the x-axis and the z-axis ($A(0,0) = I_{3\times 3}$). Note, that (4) neither depends on x_{W} nor on ϑ. A rigid body contact model defines the vertical displacement $-z_{\mathrm{W}}$ of the wheel as function of its lateral displacement y_{W} and of the angles φ, ψ:

$$z_{\mathrm{W}} = z_{\mathrm{W}}(y_{\mathrm{W}}, \varphi, \psi) \, . \tag{5}$$

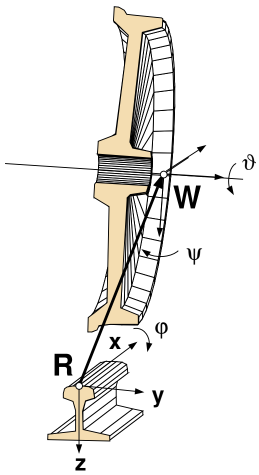

Fig. 2. Basic configuration: single wheel and single rail.

The most straightforward contact model neglects the elastic deformation totally and treats wheel and rail as undeformed geometrical bodies ([19]). Then (5) says that the wheel lies on the rail and that the minimum distance between wheel and rail vanishes. I. e. wheel and rail do not penetrate each other and they have (at least) one common point. In a common point the normal vectors to the surfaces of wheel and rail have to be parallel (otherwise wheel and rail would penetrate each other in a neighbourhood of the common point). All normal vectors to the rail surface are parallel to the (y, z)-plane since the rail is prismatic. Therefore the common point(s) belong(s) to the curve $\mathfrak{C}$ of all points P_{W} on the wheel surface where the normal vectors to the surface are parallel to the (y, z)-plane. In general this curve can be parametrized by the wheel coordinate s and we have ([14])

$$\mathfrak{C} = \mathfrak{C}(\psi) = \{ P_{\mathrm{W}}(s; \psi) \, : \, \underline{s} \le s \le \overline{s} \} \, . \tag{6}$$

Consider the special case $\psi = 0$ as an example: $\mathfrak{C}(0)$ consists of all points P_{W} on the wheel surface that are exactly below the symmetric axis of the wheel, i. e. below the centre of the wheel axle.

For simplicity the distance between wheel and rail is measured along the z-axis such that the distance of a point $P_{\mathrm{W}}(s; \psi) = (x, y, z)^T \in \mathfrak{C}(\psi)$ to the rail surface is given by the distance $G(y) - z$ between P_{W} and $(x, y, G(y))^T$ (see (3) and note, that the z-axis is oriented downwards). Because of (4) this distance $G(y) - z$ can be written as $-z_{\mathrm{W}} - \Delta(s; y_{\mathrm{W}}, \varphi, \psi)$ with the *distance function* Δ that is independent of z_{W}. The undeformed geometrical bodies of wheel and rail are in contact iff $0 = \min_s(-z_W - \Delta(s; y_W, \varphi, \psi))$, i. e.

$$z_W + \max_{\underline{s} \le s \le \overline{s}} \Delta(s; y_W, \varphi, \psi) = 0 \; . \tag{7}$$

The common points $P_W(s^*; \psi)$ of wheel and rail surface are characterized by $\Delta(s^*; y_W, \varphi, \psi) = \max_s \Delta(s; y_W, \varphi, \psi)$. There are several ways to evaluate the contact condition (7) efficiently ([19, 14]) but its main drawback is obvious: (7) defines a continuous constraint that is in general only piecewise differentiable. A typical phenomenon in the contact of rails and wheels with wear profiles are *jumps* of the common point, i. e. s^* does not vary continuously with (y_W, φ, ψ) ([14], see also Fig. 3). If s^* changes discontinuously then (7) is in general not differentiable. Eqs. (1) with contact conditions (7) do not have a (classical) solution.

In [7] several attempts have been made to avoid these difficulties. Instead of neglecting the elastic deformation totally the modified contact model neglects only the quantitative effects but considers the elastic deformation qualitatively. We replace (7) by

$$z_W + \mathrm{smax}_s^{(\nu)} \Delta(s; y_W, \varphi, \psi) = 0 \tag{8}$$

with

$$\mathrm{smax}_s^{(\nu)} \Delta(s; y_W, \varphi, \psi) := \nu \ln\Big(\frac{1}{\overline{s} - \underline{s}} \sum_{i=1}^{N} (s_i - s_{i-1}) \exp(\frac{1}{\nu} \Delta(s_i; y_W, \varphi, \psi)) \Big) \; .$$

Here $\nu > 0$ denotes a small parameter and $\{s_i\}$ with $\underline{s} = s_0 < s_1 < \ldots < s_N = \overline{s}$ a grid on $[\underline{s}, \overline{s}]$. For $\nu \to 0$ function $\mathrm{smax}_s^{(\nu)}(s; y_W, \varphi, \psi)$ converges pointwise to $\max_s \Delta(s; y_W, \varphi, \psi)$. If $\nu > 0$ then the contact condition (8) is as differentiable as the distance function Δ, i. e. we get sufficient regularity in the model equations (1). The price for this regularity is high: one evaluation of $\mathrm{smax}^{(\nu)}$ requires N evaluations of the distance function. Depending on ν this number N has to be (very) large to get a reasonable contact condition, typical values are $10^3 \ldots 10^4$.

In the following we refer to (7) and (8) as the *basic* and the *regularized* (contact) *model*, respectively. Besides the geometrical contact condition we have to define the point on the wheel surface in that constraint force and friction force operate. For historical reasons this point is called *contact point* $P_W(S)$ with $S = S(y_W, \varphi, \psi)$. In the basic model it coincides with the common point (take the one with smallest s if there are more than one common points). In the regularized model we choose S as weighted mean value of s along $\mathfrak{C}$ with weights $\exp(\frac{1}{\nu}\Delta(s; y_W, \varphi, \psi))$ (see Fig. 4).

To avoid the drawbacks of both rigid contact models we use in the dynamical simulation a weighted least squares approximation of (7) or of (8). The approximation error is kept in the range of the model error (that is $\ll 1\,\mathrm{mm}$

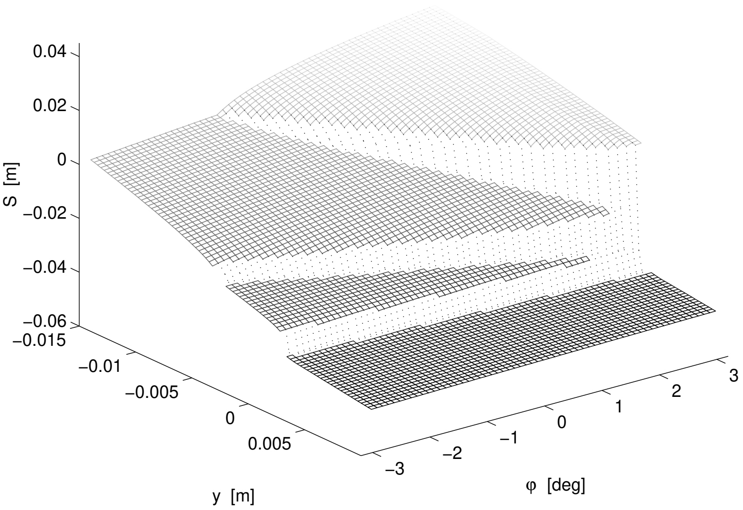

Fig. 3. Basic contact model: contact point $S(y_{\mathrm{W}}, \varphi, 1.0°)$, wheel profile S1002, rail profile UIC60–ORE.

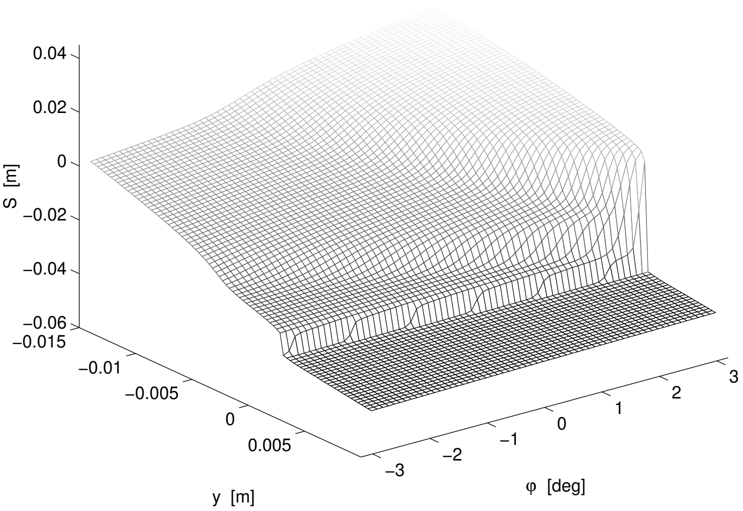

Fig. 4. Regularized contact model: contact point $S(y_{\mathrm{W}}, \varphi, 1.0°)$, $\nu = 2.0_{10}{-5}$, wheel profile S1002, rail profile UIC60–ORE.

in the geometrical contact condition). One key observation is that in comparison with y_{W} and φ the influence of the yaw angle ψ on the distance funtion Δ is very small. Furthermore, $|\psi|$ is small during simulation and Δ is an odd function w. r. t. ψ. These facts allow to decouple the 3D problem "approximate $f(y_{\mathrm{W}},\varphi,\psi)$" into several 2D approximation problems with fixed yaw angles ψ. (Here $f(y_{\mathrm{W}},\varphi,\psi) := \max_s \Delta(s;y_{\mathrm{W}},\varphi,\psi)$ in the basic model and $f(y_{\mathrm{W}},\varphi,\psi) := \mathrm{smax}_s^{(\nu)}\Delta(s;y_{\mathrm{W}},\varphi,\psi)$ in the regularized model.) Currently we use

$$f(y_{\mathrm{W}},\varphi,\psi) \approx (1-(\psi/\hat{\psi})^2)f(y_{\mathrm{W}},\varphi,0) + (\psi/\hat{\psi})^2 f(y_{\mathrm{W}},\varphi,\hat{\psi})$$

with $\hat{\psi} := 3^\circ$ and approximate $f(y_{\mathrm{W}},\varphi,0)$ and $f(y_{\mathrm{W}},\varphi,\hat{\psi})$ separately.

From the great variety of possible approximations for a function $f(y_{\mathrm{W}},\varphi,\hat{\psi})$ with fixed $\hat{\psi}$ that still depends on the 2 variables y_{W} and φ we selected a polynomial spline ([5]) since the approximating spline can be computed solving a system of *linear* equations and it does not need any a priori information about the function to be approximated. This approximation is very robust, a potential user does not need any special knowledge about the algorithm and the algorithm itself can be used for many different sets of input data, i. e. it is open to future changes in the model. On the other hand the spline approximation is flexible: approximation error and differentiability of the spline can be controlled using splines with appropriate order and grid.

The regular pattern of the data in Fig. 4 motivates the restriction to tensor-product splines on rectangular grids to simplify the computation of the spline. I. e. along each gridline in a (y_{W},φ)-grid the approximation coincides with a standard 1D spline function. It is well known that discontinuities in the data need special care in the approximation process. We therefore fit the tensor-product spline $\sigma(y_{\mathrm{W}},\varphi)$ not only to the given function f but add a smoothing term. More precisely, the spline σ is defined such that it minimizes a discrete version of the functional

$$\begin{aligned}&\iint_\Omega \omega(y_{\mathrm{W}},\varphi)\Big(\sigma(y_{\mathrm{W}},\varphi) - f(y_{\mathrm{W}},\varphi,\hat{\psi})\Big)^2 dy_{\mathrm{W}}d\varphi + \\ &+ \iint_\Omega (\mu_1^2(\frac{\partial^2\sigma}{\partial y_{\mathrm{W}}^2}(y_{\mathrm{W}},\varphi))^2 + 2\mu_1\mu_2(\frac{\partial^2\sigma}{\partial y_{\mathrm{W}}\partial\varphi}(y_{\mathrm{W}},\varphi))^2 + \mu_2^2(\frac{\partial^2\sigma}{\partial\varphi^2}(y_{\mathrm{W}},\varphi))^2)\,dy_{\mathrm{W}}d\varphi\end{aligned} \tag{9}$$

with fixed non-negative parameters $\mu_1,\mu_2 \in \mathbb{R}$ and a (positive) *weight function* $\omega(y_{\mathrm{W}},\varphi)$ that allows to introduce a priori knowledge about the given function f (see the next section for details). The smoothing term is a generalization of the Thin–Plate functional.

Example 1 In a neighbourhood of the jumps of the contact point in Fig. 3 it does not make sense to compute a polynomial spline that is close to the (discontinuous) data, i. e. $\omega(y_{\mathrm{W}},\varphi)$ should be small in these regions.

3 THE COMPUTATION OF THE APPROXIMATING 2D-TENSORPRODUCT SPLINE

Computing the 2D-tensorproduct spline we have to determine a few hundred spline coefficients on the basis of a few thousand data points $f_l = f(y_{\mathrm{W},l}, \varphi_l, \hat{\psi})$. Using a B-spline representation the spline coefficients are obtained as least squares solution of a large sparse overdetermined system of linear equations. In this section we sketch how this sparse overdetermined system can be solved efficiently. In some examples we compare the spline approximation with the original data f_l and illustrate the use of the smoothing term.

To describe the algorithm we write the spline σ as linear combination of B-splines ([5]) adopting ideas and the notation of [18]. Let $k \in \mathbb{N}$ and knot sequences $t^{(i)}$, $(i = 1, 2)$ be given with $k \geq 2$ and

$$t_1^{(i)} = \ldots = t_k^{(i)} = \underline{x}^{(i)} < t_{k+1}^{(i)} \leq \ldots \leq t_{n_i}^{(i)} < \overline{x}^{(i)} = t_{n_i+1}^{(i)} = \ldots = t_{n_i+k}^{(i)} \; .$$

The sequences $\{ B_{kj}^{(i)} \}$ of normalized B-splines of order k with respect to the knot sequences $t^{(i)}$ are recursively defined by

$$\begin{aligned} B_{1j}^{(i)}(x) &\equiv 0 \quad \text{if } j < k \text{ or } j > n_i \;, \\ B_{1j}^{(i)}(x) &= \begin{cases} 1 & \text{for } x \in I_j \;, \\ 0 & \text{otherwise,} \end{cases} \qquad (j = k, \ldots, n_i) \end{aligned}$$

with $I_j := [t_j^{(i)}, t_{j+1}^{(i)})$ for $j = k, \ldots, n_i - 1$, $I_{n_i} := [t_{n_i}^{(i)}, t_{n_i+1}^{(i)}]$ and

$$B_{kj}^{(i)} := \omega_{kj}^{(i)} B_{k-1,j}^{(i)} + (1 - \omega_{kj}^{(i)}) B_{k-1,j+1}^{(i)} \tag{10}$$

where

$$\omega_{kj}^{(i)}(x) := \begin{cases} \dfrac{x - t_j^{(i)}}{t_{j+k-1}^{(i)} - t_j^{(i)}} & \text{if } t_{j+k-1}^{(i)} - t_j^{(i)} > 0 \;, \\ 0 & \text{otherwise.} \end{cases}$$

The approximating spline is given by

$$\sigma(x^{(1)}, x^{(2)}) = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} B_{k,j_1}^{(1)}(x^{(1)}) B_{k,j_2}^{(2)}(x^{(2)}) \alpha_{j_1 j_2} \tag{11}$$

with spline coefficients $\alpha_{j_1 j_2}$.

Remark 1 a) The tensorproduct splines $\{ B_{k,j_1}^{(1)} B_{k,j_2}^{(2)} \, : \, 1 \leq j_i \leq n_i, \, i = 1, 2 \}$ are a basis of the $(n_1 \cdot n_2)$-dimensional vectorspace of all spline functions σ

defined by (11), they satisfy

$$\sum_{j_1=1}^{n_1}\sum_{j_2=1}^{n_2} B_{k,j_1}^{(1)}(x^{(1)})B_{k,j_2}^{(2)}(x^{(2)}) \equiv 1 \ , \quad (x^{(1)} \in [\underline{x}^{(1)}, \overline{x}^{(1)}],\ x^{(2)} \in [\underline{x}^{(2)}, \overline{x}^{(2)}])$$

and have finite support

$$\text{supp}\ (B_{k,j_1}^{(1)}(x^{(1)})B_{k,j_2}^{(2)}(x^{(2)})) = [t_{j_1}^{(1)}, t_{j_1+k}^{(1)}] \times [t_{j_2}^{(2)}, t_{j_2+k}^{(2)}] \ . \tag{12}$$

b) We restrict ourselves to grids with $t_k^{(i)} < t_{k+1}^{(i)} < \ldots < t_{n_i+1}^{(i)}$ to get a $(k-2)$-times continuously differentiable spline function. Typical orders are $k = 5$ for the approximation of the contact condition and $k = 3$ for the approximation of the contact point S.

c) To simplify notation we set $x^{(1)} := y_{\text{w}}$, $x^{(2)} := \varphi$ in this section. In the actual computations a slightly improved version is used, see Remark 2b.

The first term in (9) is the weighted L_2-norm $\|\sigma - f\|_{2,\omega}^2$. This term is substituted by its discrete analogue

$$\Phi_1(\sigma) := \sum_{l=1}^{m} \omega_l \Big(\sigma(x_l^{(1)}, x_l^{(2)}) - f_l\Big)^2 \tag{13}$$

with given points $(x_l^{(1)}, x_l^{(2)})$, $(l = 1, \ldots, m)$ and function values $f_l = f(x_l^{(1)}, x_l^{(2)})$ $= f(y_{\text{w},l}, \varphi_l, \hat{\psi})$ that are either evaluated during the computation of σ or read from a file. In general there are much more data points than free spline coefficients: $m \gg n_1 n_2$. To consider the distribution of data points in the $x^{(1)}$-$x^{(2)}$-plane we define for each rectangle

$$R_{j_1 j_2} = [t_{j_1+k-1}^{(1)}, t_{j_1+k}^{(1)}) \times [t_{j_2+k-1}^{(2)}, t_{j_2+k}^{(2)})$$

the number $N_{j_1 j_2}$ of data points in $R_{j_1 j_2}$ and the area $A_{j_1 j_2}$ of $R_{j_1 j_2}$. If $(x_l^{(1)}, x_l^{(2)}) \in R_{j_1 j_2}$ then we use the weight

$$\omega_l = \frac{1}{N_{j_1 j_2}} \omega(y_{\text{w},l}, \varphi_l) \cdot A_{j_1 j_2} \ .$$

Inserting (11) function $\Phi_1(\sigma)$ in (13) can be written as

$$\Phi_1(\sigma) = \sum\nolimits_l \omega_l \Big(\sum\nolimits_{j_1,j_2} B_{k,j_1}^{(1)}(x_l^{(1)})B_{k,j_2}^{(2)}(x_l^{(2)})\alpha_{j_1 j_2} - f_l\Big)^2 = \|B\alpha - d\|_2^2$$

with the unknown spline coefficients

$$\alpha = (\, \alpha_{11}, \alpha_{12}, \ldots, \alpha_{1n_2}, \alpha_{21}, \ldots, \alpha_{n_1 n_2}\,)^T \in \mathbb{R}^{n_1 n_2} \ , \tag{14}$$

a matrix $B \in \mathbb{R}^{m\times(n_1n_2)}$ and a vector $d \in \mathbb{R}^m$, that contains the (weighted) values f_l. Each data point $(x_l^{(1)}, x_l^{(2)})$ defines one row of B and d. Because of (12) at most k^2 terms in $\sigma(x^{(1)}, x^{(2)}) = \sum_{j_1,j_2} B^{(1)}_{k,j_1}(x^{(1)})B^{(2)}_{k,j_2}(x^{(2)})\alpha_{j_1j_2}$ do not vanish. I. e. each row of matrix B has (at most) k^2 non-zero elements, all rows corresponding to data points in one and the same rectangle $R_{j_1j_2}$ have the same sparsity pattern. This gives matrix B a very regular sparsity structure that can be exploited in the computation of the spline coefficients α.

More precisely the set of all row vectors of B that correspond to data points in a rectangle $R_{j_1j_2}$ can be written as a matrix $\Gamma_{j_1j_2} \in \mathbb{R}^{N_{j_1j_2}\times(n_1n_2)}$ that has (at most) k^2 non-zero columns. Collecting these non-zero column vectors in a matrix $C_{j_1j_2} \in \mathbb{R}^{N_{j_1j_2}\times k^2}$ matrix $\Gamma_{j_1j_2}$ can be written as matrix product $\Gamma_{j_1j_2} = C_{j_1j_2}P_{j_1j_2}$ with the sparse matrix

$$P_{j_1j_2} = \begin{pmatrix} 0_{k\times(j_1-1)} & I_k & 0_{k\times(n_1+1-k-j_1)} \end{pmatrix} \otimes \begin{pmatrix} 0_{k\times(j_2-1)} & I_k & 0_{k\times(n_2+1-k-j_2)} \end{pmatrix},$$

$P_{j_1j_2} \in \mathbb{R}^{k^2\times(n_1n_2)}$. Here $0_{k\times j} \in \mathbb{R}^{k\times j}$ denotes the zero matrix, $I_k \in \mathbb{R}^{k\times k}$ the identity matrix and $\otimes$ the Kronecker product of matrices.

To solve $\|B\alpha - d\|_2^2 \to \min$ for α we transform the overdetermined system of linear equations $B\alpha = d$ to

$$\begin{pmatrix} \hat{B} \\ 0 \end{pmatrix} \alpha = \hat{d} \tag{15}$$

with an upper triangular matrix $\hat{B} \in \mathbb{R}^{(n_1n_2)\times(n_1n_2)}$ using orthogonal transformations.

The algorithm consists of two parts: **Algorithm I** computes for one single rectangle $R_{j_1j_2}$ a QR-factorization of $C_{j_1j_2}$ ([9]), (fast) Givens rotations are applied to transform $C_{j_1j_2}$ to $\binom{\hat{C}_{j_1j_2}}{0} \in \mathbb{R}^{N_{j_1j_2}\times k^2}$ with an upper triangular matrix $\hat{C}_{j_1j_2} \in \mathbb{R}^{k^2\times k^2}$. This Algorithm I is applied consecutively to all rectangles of the mesh such that matrix B is transformed to $\binom{\tilde{B}}{0}$ with a matrix $\tilde{B}$ that consists of all the non-zero rows of the matrices $\hat{C}_{j_1j_2}P_{j_1j_2}$.

The transformation of $B\alpha = d$ to (15) is completed by **Algorithm II** applying (fast) Givens rotations to $\tilde{B}$. The main difference between both algorithms is the fact that Algorithm I works with vectors of length $\leq k^2$ and Algorithm II processes vectors with up to $n_2(k-1)+k$ non-zero elements.

Using these two algorithms matrix $\hat{B}$ and vector $\hat{d}$ in (15) can be obtained efficiently and with low storage requirements. Matrix $\hat{B}$ is an upper triangular band matrix with bandwidth $n_2(k-1)+k$. For a more detailed discussion of the algorithm we refer to the technical report [1].

The partial derivatives of σ in the second term of (9) can again be expressed by B-splines. Using a suitable approximation of the integrals the smoothing term in (9) may be substituted by a term of the form

$$\Phi_2(\sigma) = \mu_1^2 \|F_{20} D_{20} \alpha\|_2^2 + 2\mu_1\mu_2 \|F_{11} D_{11} \alpha\|_2^2 + \mu_2^2 \|F_{02} D_{02} \alpha\|_2^2$$

with large sparse matrices $F_{r_1 r_2}$ and $D_{r_1 r_2}$. This approximation of the smoothing term is used in 1D e. g., in [18], for the details of the 2D-version we again refer to [1].

Taking into account the smoothing term the coefficients α are defined by $\Phi_1(\sigma) + \Phi_2(\sigma) \to \min$ and the overdetermined system $B\alpha = d$ has to be extended by additional equations. α is the least squares solution of the overdetermined system

$$\begin{pmatrix} B \\ \mu_1 F_{20} D_{20} \\ \sqrt{2\mu_1\mu_2} F_{11} D_{11} \\ \mu_2 F_{02} D_{02} \end{pmatrix} \alpha = \begin{pmatrix} d \\ 0 \\ 0 \\ 0 \end{pmatrix} . \tag{16}$$

If the smoothing term does not vanish and $\mu_1 \neq 0$, $\mu_2 \neq 0$ then the unique solvability of the least squares problem $\Phi_1(\sigma) + \Phi_2(\sigma) \to \min$ is guaranteed under mild assumptions, e. g., if there is at least one data point per rectangle $R_{j_1 j_2}$. The smoothing term $\Phi_2(\sigma)$ introduces $3n_1 n_2 - 3(n_1 + n_2) + 1$ additional equations in (16). The sparsity structure of the matrices $D_{r_1 r_2}$ and $F_{r_1 r_2}$ allows to assign each of these equations in a natural way to one of the rectangles $R_{j_1 j_2}$, i. e. the additional equations can be processed by a modification of Algorithm I (with 3 additional equations in most of the rectangles).

In this way (16) is transformed to a system with upper triangular matrix of coefficients, the solution α is obtained by a back-substitution ([9, Chapter 5]). The application of orthogonal transformations and the B-spline representation make the numerical solution robust ([9], [5]). The algorithm is not sensitive w. r. t. failures in the evaluation of function values f_l in (13). If for any reason function f can not be evaluated at some $(x_1^{(l)}, x_2^{(l)})$ then the corresponding row in B and d is simply deleted. Once the spline coefficients α are computed the spline σ can be evaluated efficiently and in a numerically stable way from (10) and (11).

We end this part with some examples. In the tests we use the same coordinates $(x^{(1)}, x^{(2)})$, the same type of grids and the same weight function as in the Pre-Processor of the simulation software (see Remark 2 for more details). To compare spline and given data we prefer 2D plots since they show more details. In the plots of Figs. 5–7 the roll angle φ is fixed to $\varphi = 1.43°$ corresponding to the nominal position of a rigid wheelset on a straight track. The splines are computed on a grid with 20×10 rectangles, the grid in $x^{(2)}$-direction is equidistant, the grid in $x^{(1)}$-direction is based on an equidistant grid with 15 intervals (see Remark 2).

Fig. 5 shows for $\psi = 0$ the vertical displacement computed from the basic and the regularized contact model, and the spline approximating the basic contact model (for the regularized contact model given data and spline coincide in the plot accuracy). The jumps of the common point, i. e. the discontinuities of the basic model, are marked by circles. The 2D-tensorproduct spline with $\mu_1 = \mu_2 = 10^{-8}$ and $k = 5$ is three times continuously differentiable, the approximation error remains in the size of the model error of the rigid contact models.

Because of discontinuities in the basic model and derivatives of large absolute value in the regularized model the smoothing term is extremely important. This is nicely illustrated by Figs. 6 and 7 that show for $\psi = 1.0°$ the contact point and its spline approximation with order $k = 3$ once for the basic and once for the regularized contact model. Without the smoothing term the approximation oscillates and can not be used in the dynamical simulation. Adding the smoothing term with $\mu_1 = \mu_2 = 10^{-6}$ a rather good approximation of the contact point is obtained. These results illustrate that the 2D-tensorproduct spline gives really useful approximations for the geometrical data of wheel-rail contact.

4 EXPLOITING PARALLELISM IN THE COMPUTATION OF THE SPLINE COEFFICIENTS

After the tensorproduct spline has been computed it can be used in the dynamical simulation again and again if wheel and rail profile do not change. So it is not too important to get the spline coefficients extremely fast, cpu-times in the range of minutes are acceptable. Nevertheless the numerical effort is large if in (13) the evaluation of f_l is very time-consuming (e. g., large N in (8)) and / or the number m of data points is large. To keep cpu-time small we implemented a parallel version of the proposed algorithm on a cluster of workstations. In this section we report on the performance on a homogeneous and on a heterogeneous network.

Computing the approximating spline the overall cpu-time covers both the costs for m function evaluations f_l in (13) and the costs for the solution of the overdetermined system (16). There are three main facts that might result in a large numerical effort: extremely fine meshs (i. e. large n_1, n_2), many data points (i. e. large m), and time consuming function evaluations f_l. From a practical point of view we can restrict ourselves to meshs of moderate size (typical values are $n_1 = 15 \ldots 30$, $n_2 = 10 \ldots 20$). Extremely fine meshs would

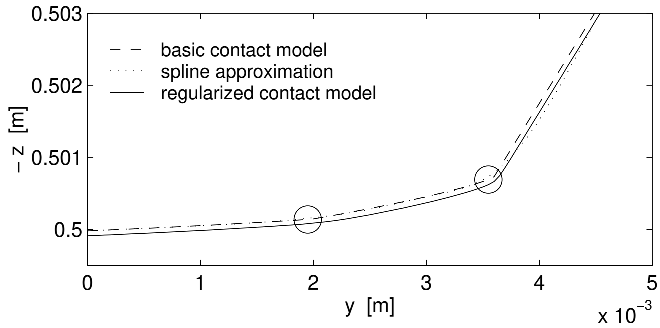

Fig. 5. Contact model: basic model, regularized model, and spline approximation.

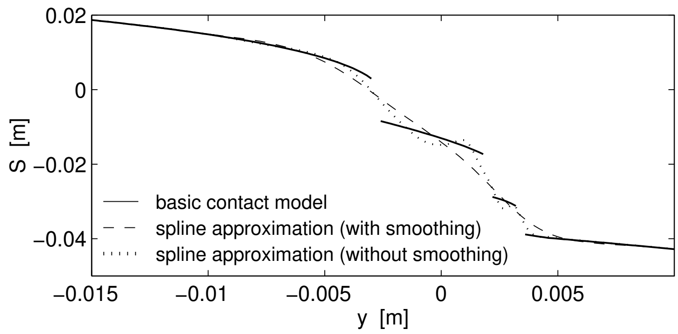

Fig. 6. Basic contact model ($\varphi = 1.43°$, $\psi = 1.0°$): approximation of contact point S.

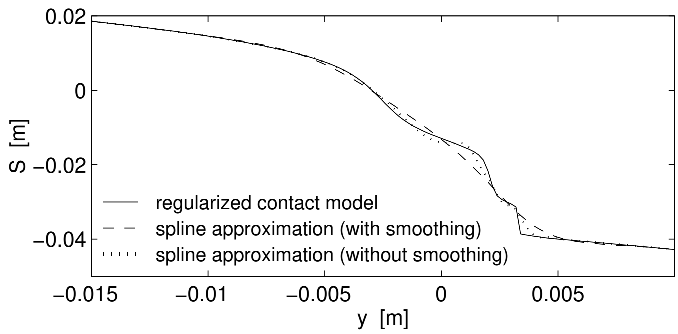

Fig. 7. Regularized contact model ($\varphi = 1.43°$, $\psi = 1.0°$): approximation of contact point S.

decrease the efficiency of the simulation software since large arrays of spline coefficients would have to be handled in the simulation package. Looking at the real physical system there is no need for such fine meshs since they could reduce only the approximation error in (9) but not the model error of the rigid contact model.

The linear algebra part of the costs could be reduced drastically if we would restrict ourselves to weights $\omega_l = \omega^{(1)}(x^{(1)}) \cdot \omega^{(2)}(x^{(2)})$ and to data points $(x_1^{(l)}, x_2^{(l)})$ on a rectangular mesh in (13) since matrix B in (16) has in this special case a tensorproduct structure that can be exploited in the solution process (see e. g., [6, Chapter 10]). In our application the restriction to special weight functions is not acceptable. Furthermore the savings in the *overall* costs are small if the costs for the evaluation of f_l dominate. For both reasons we do not follow this mesh data approach.

Instead of reducing the numerical effort for the linear algebra part we distribute the whole work among a few workstations that are coupled to a cluster via Internet. The algorithm of Section 3 consists of the sequential part (Algorithm II) and of Algorithm I that can be applied in parallel to the $(n_1 - k + 1)(n_2 - k + 1)$ rectangles of the mesh. This motivates a master–slave model of parallelization. In a cluster of n_{host} workstations one master does the sequential work and sends jobs to $n_{\text{host}} - 1$ slaves. All these "jobs" have identical structure: for a given rectangle $R_{j_1 j_2}$ the slave evaluates the $N_{j_1 j_2}$ function values f_l and applies Algorithm I to get the upper triangular scheme $\hat{C}_{j_1 j_2}$ that is sent back to the master.

It is hard to predict the actual performance of a slave process in a multi-user workstation environment. That is why we use a dynamic load balance scheme, i. e. fast slaves get more jobs than slow ones ([8, Section 8.3]). This load balance scheme is discussed in detail in Section 4 of the technical report [1].

The master–slave model was implemented using PVM ([8]). The sequential code was extended by some subroutines that handle the dynamic load balance scheme and the communication between master and slaves. The communication costs of the parallel algorithm are small, the master sends only scalars to the slaves $(j_1,\ j_2,\ \ldots)$ and receives matrices $\hat{C}_{j_1 j_2}$ of size $k^2 \times k^2$. To rate the performance of the parallel algorithm we compare the sequential program with two parallel versions that differ by the hardware in use. The sequential algorithm runs on a SUN Sparc5 that is also used as master in the parallel version. We couple up to three additional Sparc5 stations with identical hard- and software to this master to get the *homogeneous* network. At the weekend we have more resources available and include two HP workstations to the network (*heterogeneous* network). We expect one of them (hp 735/99) to be much faster than the Sparc5's, the other one is older (hp 710) and has a hard job as `ftp`–server, it represents a potential bottleneck in our network.

The parallelization was mainly motivated by time-consuming function evaluations f_l, in future models f_l could e. g., even contain solutions of integral equations. If the mesh is of moderate size and $m \gg n_1 n_2$ then both the linear algebra part of the costs and the costs for m function evaluations f_l grow linearly in m ([1, Remark 2]). To simplify the quantitative comparison of the algorithms we therefore fix the function f that is to be approximated, and increase m, i. e. the number of data points. In the following numerical tests the contact condition (8) with $N = 200$ is approximated on a mesh with 20×10 rectangles. We chose the number m of data points equal to the number of equations in (16), i. e. in this comparison the (small) additional effort for the smoothing term is not considered.

Example 2 Figs. 8 and 9 show for varying m the cpu-time of the sequential algorithm ("$\times$"), of the parallel algorithm on the homogeneous network with one ("$\circ$"), two ("$+$"), and three ("$*$") slaves, respectively, and of the parallel algorithm on the heterogeneous network with 5 slaves ("$\otimes$"). The figures differ by the order k of the spline: we have $k = 5$ (i. e. $n_1 = 24$, $n_2 = 14$) in Fig. 8 and $k = 3$ (i. e. $n_1 = 22$, $n_2 = 12$) in Fig. 9. The bandwidths of the resulting upper triangular systems (15) are 61 and 27, respectively. In both cases the benefits of the parallelization are obvious for large m. For small m we do not expect a substantial speed-up by parallelization because the sequential part of the linear algebra costs dominates the overall costs. Table 1 corresponds to Fig. 8 and summarizes the speed-up of the parallel algorithm. Asymptotically for $m \to \infty$ a speed-up of $\approx n_{\text{host}} - 1$ is reached on the homogeneous network of n_{host} workstations (i. e. with $n_{\text{host}} - 1$ slaves).
Table 1 and Fig. 8 show nicely the advantages of a homogeneous network that equidistributes the jobs among the slaves (i. e. approximately 67 jobs per slave in the network with 3 slaves). The heterogeneous network results in different loads for the slaves (a typical load balance are 37, 38, and 39 jobs on the Sparc5's, 59 jobs on the hp 735/99, and 29 jobs on the hp 710). Compared with the homogeneous network without the HP's a substantial speed-up is obtained for $m \geq 3000$, only. For very large m, however, it is clearly superior (see also Fig. 9).
For smaller spline order parallelization reduces cpu-time already for small m. Compared with Fig. 8 we have in Fig. 9 less sequential work (because of the smaller bandwidth) and less communication costs ($k^2 = 9$ instead of $k^2 = 25$). For large m both figures are almost identical since linear algebra and communication costs are dominated by the m function evaluations f_l.

This example illustrates that the proposed algorithm works satisfactorily in a typical workstation environment. For time-consuming function evaluations f_l the speed-up is proportional to the number of slaves that are added to the network. We want to point out that neither the sequential nor the parallel

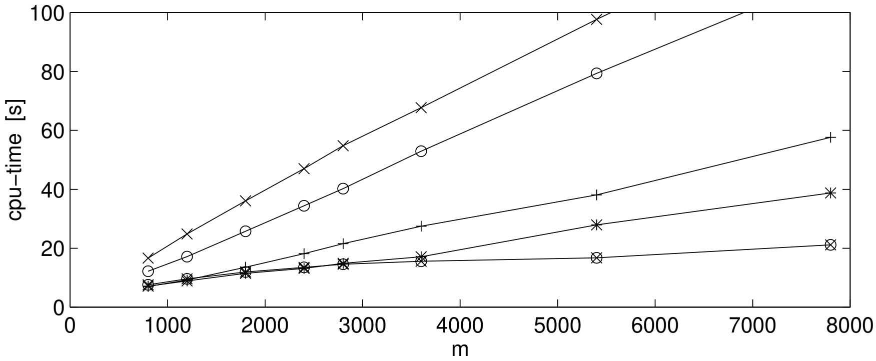

Fig. 8. Numerical effort of sequential and parallel algorithms, $k = 5$, $n_1 = 24$, $n_2 = 14$.

m	800	1200	1800	2400	2800	3600	5400	7800
1 slave	1.37	1.45	1.40	1.37	1.36	1.28	1.23	1.21
2 slaves	2.37	2.71	2.66	2.59	2.54	2.46	2.56	2.35
3 slaves	2.34	2.79	3.14	3.58	3.68	3.96	3.50	3.50
5 slaves	2.19	2.59	3.03	3.48	3.77	4.34	5.83	6.41

Table 1. Speed-up of parallel algorithms, $k = 5$, $n_1 = 24$, $n_2 = 14$.

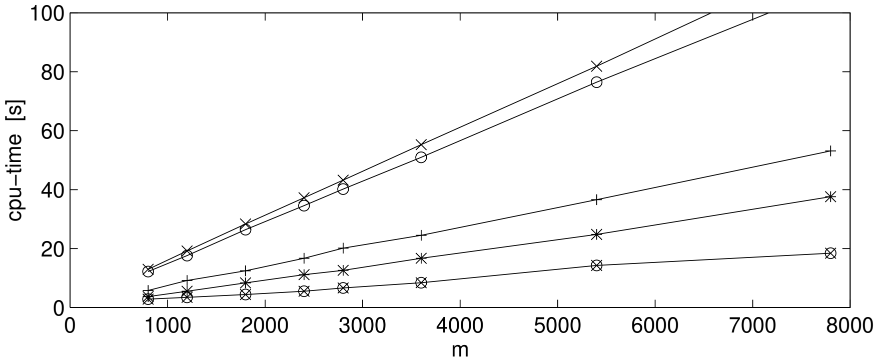

Fig. 9. Numerical effort of sequential and parallel algorithms, $k = 3$, $n_1 = 22$, $n_2 = 12$.

algorithm is iterative. Except round-off errors the results of both algorithms are identical.

5 APPROXIMATED CONTACT CONDITIONS IN THE DYNAMICAL SIMULATION

As part of the simulation software the approximation of contact conditions allows the efficient dynamical simulation of wheel-rail systems. In this section we discuss the interaction of approximation and simulation. Simulation results for a simple prototype (rigid wheelset on a straight track) illustrate the efficiency of the proposed algorithm. For the application to real-life problems in railway vehicle industry we refer to [17].

The basic component of a simulation package is the numerical integration of model equations. Advanced packages like SIMPACK combine this basic task with powerful pre- and post-processing facilities to build up the models and to analyse the simulation results ([17]). The SIMPACK user has to specify wheel and rail profile. During the integration of (1) the contact condition (8) has to be evaluated several thousand times (*on-line regularization*). If approximated contact conditions are used then the coefficients of the approximating splines are computed within the pre-processing step. Using the spline approximation the numerical effort is reduced substituting the contact condition by the corresponding spline (*off-line regularization*).

In the present paper we focus on the formulation and on the approximation of the geometrical contact conditions. In the dynamical simulation we need additionally a model for the friction forces between wheel and rail. The friction forces depend mainly on the normal forces, on the slip and on the stress distribution in the contact patch. Various models are known from the literature, between these models there are extremely large differences in the model error and in the numerical effort that is necessary to compute the friction forces (for a detailed discussion see [11]). The regularized contact model might be coupled with virtually any of these models. In Examples 3 and 4 we use a generalized Hertzian contact model ([2]) to get the distribution of the stresses in normal direction and apply afterward Kalker's code FASTSIM ([11]) to compute the friction forces between wheel and rail.

The Hertzian contact model is based on an approximation of the surfaces of wheel and rail by paraboloids. This approximation allows the efficient computation of the distribution of the normal stresses in the contact patch (see e. g., [10, Chapter 4]). In the classical model a common point $P_{\mathrm{w}}(s^*) = P_{\mathrm{r}}(P_{\mathrm{w}}(s^*))$ is chosen as apex of the paraboloids and the paraboloids are defined as 2nd order approximation of the two surfaces in the common point $P_{\mathrm{w}}(s^*) = P_{\mathrm{r}}(P_{\mathrm{w}}(s^*))$,

i. e. in the common point the tangential plane and the curvatures of the first paraboloid coincide with the tangential plane and the curvatures of the wheel surface and the ones of the second paraboloid with that of the rail surface. For wheels with wear profiles this classical Hertzian model results in a large model error ([12], see also Example 4 below).

In [2] we proposed a generalization of the Hertzian contact model that uses again paraboloids as approximation of the surfaces such that the distribution of the normal stresses can be obtained as before. However, instead of the common point $P_{\rm W}(s^*)$ we choose the contact point $P_{\rm W}(S)$ of the regularized contact model as apex of the paraboloids. In $P_{\rm W}(S)$ the tangential planes of the paraboloids coincide with the ones of the wheel and rail surface, respectively. Weighted mean values of the curvatures of wheel and rail surface define the curvatures of the paraboloids in their apex. These mean values are computed along curve $\mathfrak{C}$ with weights $\exp(\frac{1}{\nu}\Delta(s; y_{\rm W}, \varphi, \psi))$ (similar to the definition of the contact point $P_{\rm W}(S)$ in Section 2).

This generalized Hertzian contact model is described in detail in [13], it is related to the equivalent Hertzian contact patches that are used in [16]. The combination of FASTSIM and the generalized Hertzian contact model computes the friction forces efficiently and — as long as the contact patch between wheel and rail is simply connected — with an accuracy that is acceptable for many applications ([13]). The computation of the curvatures as weighted mean values along $\mathfrak{C}$ is cpu-time consuming. Therefore we approximate these data again by tensorproduct splines.

The wheel-rail contact model is only one modul of the MBS model. So we have to avoid complicated data structures and extremely large data arrays in the off-line regularization. Spline approximations are used only in model components that can not be evaluated efficiently on-line. These are the (regularized) geometrical contact condition (8), the contact point S, and all parameters in the computation of friction forces that depend on the curvatures of wheel and rail surface.

Remark 2 a) The approximation of the contact condition is part of the constraints $g(q, t) = 0$ in (1) and has to be $r+2$ times differentiable to get a r times differentiable right hand side in (2). Therefore the contact condition is approximated by a spline of order $k^* + 2$ and all other parameters (including S) by splines of order k^*, in the actual computations we set $k^* := 3$. All the splines of order k^* are defined on one and the same mesh (i. e. with the same B-splines $B_{kj}^{(i)}$ in (10)) to save cpu-time.

b) The tensorproduct spline was defined such that the approximation gives reasonable results even if it is applied straightforwardly with coordinates $(x^{(1)}, x^{(2)})$ $= (y_{\rm W}, \varphi)$ on an equidistant mesh. However, in the simulation package we tune coordinates and meshs to the standard model of wheels with flange and tread.

If the flange gets in contact with the rails (at $y_{\mathrm{W}} \approx 3.5\,\mathrm{mm}$ in Figs. 5 and 6) the wheel starts to climb upwards immediately and we have rapid changes in the contact conditions (7) and (8). The jumps of the contact point S from tread to flange of the wheel define a curve in the $(y_{\mathrm{W}}, \varphi)$–plane that can be approximated by a straight line being parallel to the φ-axis in Fig. 3 ($y_{\mathrm{W}} \approx 3.5\,\mathrm{mm}$). Coordinates $(x^{(1)}, x^{(2)})$ are defined by a slight rotation of the $(y_{\mathrm{W}}, \varphi)$–system such that this line is given by $x^{(1)} = x_*^{(1)}$, i. e. it is parallel to the $x^{(2)}$-axis, $x^{(1)} \approx y_{\mathrm{W}}$, $x^{(2)} \approx \varphi$. The mesh should have more knots in $x^{(1)}$- than in $x^{(2)}$-direction ($n_1 \gg n_2$) since changes of the geometrical data are much less drastic in φ-direction (see Figs. 3 and 4). Starting with an equidistant mesh a few knots in $x^{(1)}$-direction are added in a neighbourhood of $x_*^{(1)}$ to consider the rapid changes of data in this region.

c) After extensive numerical tests the order k of the splines and the dimension $n_1 n_2$ of the spline space are chosen such that the approximation is close to the given data and can nevertheless be evaluated efficiently during the dynamical simulation. We want to keep the data arrays small (using meshs of moderate size i. e. spline spaces of moderate dimensions), excessive work in (10) is avoided if k is not too large. Splines with $n_1 \geq 16 + k$, $n_2 \geq 8 + k$ give reasonable results in the dynamical simulation for wheels with wear profile S1002 and rails with profile UIC60–ORE (both are industrial standards). In the actual computations of Examples 3 and 4 we use $n_1 = 19 + k$, $n_2 = 9 + k$ with $k = k^* + 2$ and $k = k^*$, respectively ($k^* = 3$), i. e. 20×10 rectangles in the mesh.

We illustrate the performance of the rigid contact models with results for 2 benchmark problems that consider the motion of a rigid wheelset on a straight track. For both benchmarks the flanges of the wheels do not get in contact with the rails. The basic contact model (7) introduces for both benchmarks discontinuities in the solution ([14], see also Example 4). We compare on-line and off-line regularization for the regularized contact model (8) with $\nu = 2.0_{10} - 5$, ($s_i - s_{i-1} \approx 0.5\,\mathrm{mm}$ at the tread, $s_i - s_{i-1} \approx 0.2\,\mathrm{mm}$ at the flange of the wheel). The model equations (1) are integrated in Gear–Gupta–Leimkuhler formulation using the BDF code DASSL ([3]).

Example 3 It is a well known phenomenon that the motion of a rigid wheelset at a straight track gets unstable if its speed exceeds a so-called critical speed. The quasi-periodic solution approaches a limit cycle. Fig. 10 shows the phase plot $(y(t), \dot{y}(t))$ for the lateral displacement y of the wheelset, $y(t_0) = 4.5\,\mathrm{mm}$. The left plot corresponds to the on-line evaluation of (8), the right one to the spline approximation. In the real physical system this limit cycle is typical and robust against changes of system parameters. The solution trajectory, however, is very sensitive w. r. t. small changes in the system. Both simulation results reproduce the limit cycle and are in good agreement with measured data. They differ, however, in the trajectories: with approximated contact conditions (right plot in Fig. 10) the solution approaches the limit cycle substantially faster. Varying the regularization parameter ν in (8) we found that this difference in the local behaviour of the solution is in the range of the model error that is caused by the rigid contact model. From the point of view of dynamical simulation the off-line regularization gives satisfactory results. Its main benefit is a reduction of cpu-time by a factor of 11. In Fig. 10 the motion was simulated on a time interval of 10.0 s with 624.8 s cpu-time for the on-line regularization and 56.3 s cpu-time for the simulation using the approximated contact conditions.

Example 4 In simulation packages it has been popular to consider only the local contact geometry such that the classical Hertzian contact model could be applied. This simplifies substantially the computation of friction forces but results in large model errors in the neighbourhood of jumps of the contact point in Fig. 3. A benchmark in [15] was constructed such that these models fail totally. A force $F_y = 20\,\mathrm{kN}$ acting in lateral direction (see Fig. 1) shifts the wheelset to the right ($y \approx 5.0\,\mathrm{mm}$). The motion tends to a quasi-stationary state that is at the right wheel very close to a configuration for that the contact point in the basic contact model (7) jumps. In [14] we applied the basic contact model to this benchmark. The simulation results in Fig. 11 illustrate the large model error of the basic contact model: The jumps of the contact point introduce discontinuities in the constraint forces (right plot) and artificial oscillations in the position coordinates (left plot).
This benchmark underlines the need for more sophisticated contact models. If we apply the regularized contact model (8) then the numerical solution reaches both for on-line and off-line regularization this quasi-stationary state, Fig. 12 shows in the left plot the vertical displacement of the wheelset and in the right plot the constraint force on the right wheel. The differences between the simulation results with on-line and with off-line regularization are negligible. Using the spline approximation the cpu-time is, however, reduced from 497.1 s to 42.4 s, i. e. by the factor 12.

The drastic reduction of cpu-time makes approximated contact conditions very attractive for simulation packages. In SIMPACK approximation is used as default contact model whenever it is applicable. The on-line evaluation of the regularized contact condition (8) is restricted to simulations with rail profiles that vary along the track (e. g., a wheelset running through a switch).

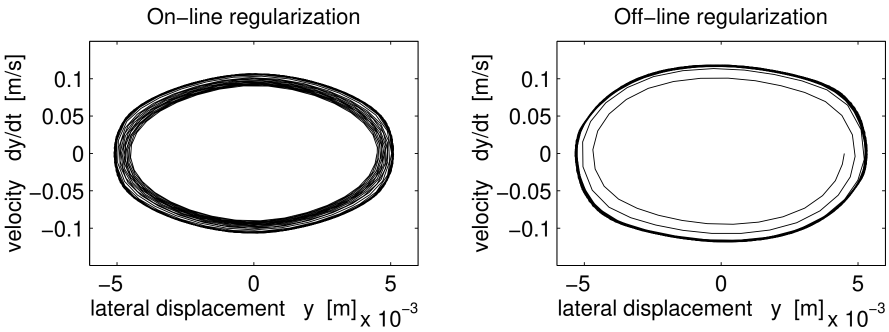

Fig. 10. Rigid wheelset moving on a straight track. Limit cycles computed with on-line (left) and off-line (right) regularization.

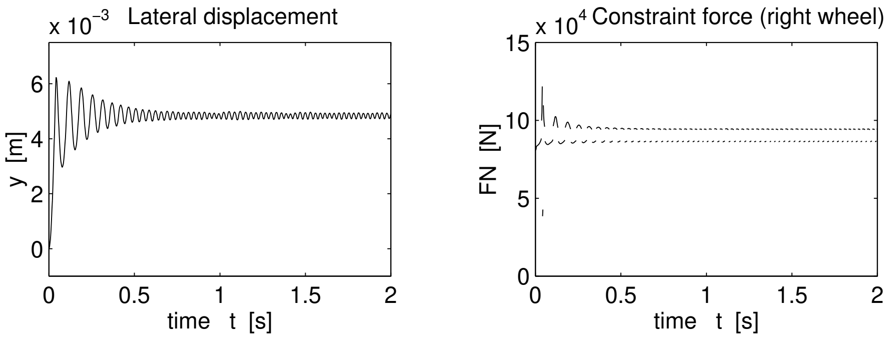

Fig. 11. Simulation results for the basic contact model applied to the benchmark of Pascal ([15]).

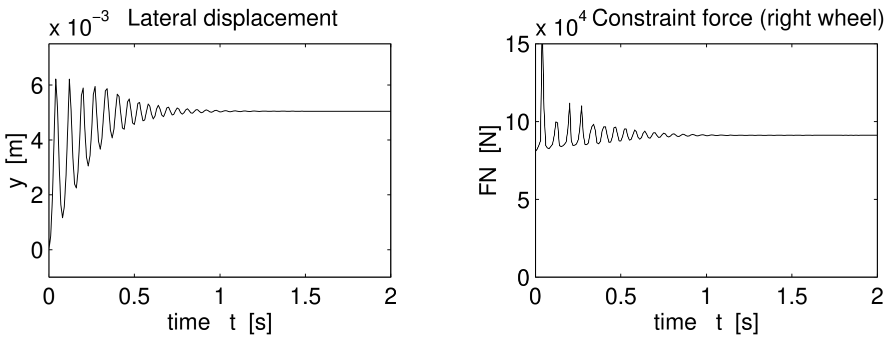

Fig. 12. Simulation results for the regularized contact model applied to the benchmark of Pascal ([15]).

6 SUMMARY

The complicated geometry of the contact between wheels with wear profiles and rails causes severe problems in the dynamical simulation of wheel-rail systems. Realistic models for contact geometry and friction forces yield to a very large numerical effort in the integration of the model equations. For a great variety of standard problems the dynamical simulation can be accelerated by a factor of 10 using *approximations* of geometrical data during the dynamical simulation. We propose a tensorproduct spline approximation that is computed in the pre-processing of the integration. Using orthogonal transformations the spline coefficients are obtained as least squares solution of a large sparse overdetermined system of linear equations. The numerical effort in the pre-processing is kept small exploiting the sparsity structure of this system, further improvements are achieved by parallelization. If the spline approximation is used then benchmark problems for simple prototypes and real-life applications from railway industry are solved satisfactorily.

The computation of the tensorproduct spline was implemented in FORTRAN. The sequential version of this code WRSPL (**W**heel–**R**ail–**SPL**ine) is available freely for non-commercial use and can be obtained contacting the first author. It is integrated in the commercial MBS simulation package SIMPACK that is tailored to industrial applications.

ACKNOWLEDGEMENTS

Part of this work has been done within the joint project "Differential equations and singular manifolds in the dynamical simulation of wheel-rail systems" of Rostock University and DLR Oberpfaffenhofen that was supported by the German Federal Minister for Education, Science, Research, and Technology (grant FR7ROK). We thank our colleagues in the wheel-rail groups in Rostock and Oberpfaffenhofen for very stimulating discussions about the topic of the present paper. The authors are especially grateful to Dirk Langemann (Rostock) for his help in the implementation of the smoothing term.

REFERENCES

1. M. Arnold and H. Netter. The approximation of contact conditions in the dynamical simulation of wheel-rail systems. Technical Report IB 515–96–08, German Aerospace Research Establishment, 1996.
2. M. Arnold and H. Netter. Wear profiles and the dynamical simulation of wheel-rail systems. to appear in *Proceedings of the 9th Conference of the European Consortium for Mathematics in Industry. Progress in Industrial Mathematics, vol. 2.* Teubner and Wiley.

3. K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical solution of initial–value problems in differential–algebraic equations.* SIAM, Philadelphia, 2nd edition, 1996.
4. N.K. Cooperrider. The hunting behaviour of conventional railway trucks. *ASME J. Eng. Industry*, 94:752–762, 1972.
5. C. de Boor. *A practical guide to splines.* Springer-Verlag, New York, 1978.
6. P. Dierckx. *Curve and surface fitting with splines.* Clarendon Press, Oxford New York Tokyo, 1993.
7. K. Frischmuth, M. Arnold, M. Hänler, and H. Netter. Differentialgleichungen und singuläre Mannigfaltigkeiten in der dynamischen Simulation von Rad–Schiene–Systemen. In K.-H. Hoffmann, W. Jäger, Th. Lohmann, and H. Schunck, editors, *Mathematik – Schlüsseltechnologie für die Zukunft*, pages 331–342. Springer-Verlag, Berlin Heidelberg New York, 1997.
8. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM3 User's Guide and Reference Manual. Technical Report ORNL/TM–12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 1994.
9. G.H. Golub and Ch.F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore London, 2nd edition, 1989.
10. K.L. Johnson. *Contact Mechanics.* Cambridge University Press, 1985.
11. J.J. Kalker. *Three-Dimensional Elastic Bodies in Rolling Contact.* Kluwer Academic Publishers, Dordrecht Boston London, 1990.
12. K. Knothe and H. Le The. A contribution to the calculation of the contact stress distribution between two elastic bodies of revolution with non-elliptic contact area. *Computers & Structures*, 18:1025–1033, 1984.
13. H. Netter. Rad–Schiene–Systeme in differential-algebraischer Darstellung. submitted, 1997.
14. H. Netter and M. Arnold. Geometrie und Dynamik eines Rad-Schiene-Modells in Deskriptorform mit unstetigen Zustandsgrößen. Technical Report IB 515–93–02, DLR, D-5000 Köln 90, 1993.
15. J.P. Pascal. Benchmark to test wheel/rail contact forces. Technical report, INRETS Paris, 1990.
16. J.P. Pascal and G. Sauvage. New method for reducing the multicontact wheel/rail problem to one equivalent rigid contact patch. In *The Dynamics of Vehicles on Roads and on Railway Tracks, 12th IAVSD–Symposium Lyon*, pages 475–489. Swets & Zeitlinger, B.V. Lisse, 1991.
17. W. Rulka, A. Haigermoser, L. Mauer, and H. Netter. Anwendung moderner Auslegungsstrategien für Schienenfahrzeuge durch Einsatz des Simulationsprogramms SIMPACK. VDI Berichte Nr. 1219, Düsseldorf, 1995.
18. H. Schwetlick and V. Kunert. Spline smoothing under constraints on derivatives. *BIT*, 33:512–528, 1993.
19. B. Simeon, C. Führer, and P. Rentrop. Differential-algebraic equations in vehicle system dynamics. *Surveys on Mathematics for Industry*, 1:1–37, 1991.