# Using Gauss-Seidel for Multibody Problems

Claude Lacoursière
HPC2N/UMIT
and
Department of Computing Science
Umeå University
SE-901 87, Umeå, Sweden
claude@hpc2n.umu.se

December 1, 2011

**Abstract**

## 1 Introduction

Discrete time-stepping a constrained multibody system involves the solution of linear systems of equations of the form

$$\begin{bmatrix} M & -G^T \\ G & \Sigma \end{bmatrix} \begin{bmatrix} v \\ \lambda \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix}. \tag{1}$$

Here, $M$ is a square $n \times n$ real, symmetric, positive definite matrix that is assumed to be easy to invert being block diagonal, and each block being small. Matrix $G$ is real and has dimensions $m \times n$ where, usually $m \leq n$, though this is not strictly required as long as the diagonal, nonnegative matrix $\Sigma$, with dimensions $m \times m$, has strictly positive diagonal entries, i.e., $\Sigma = \mathrm{diag}(\epsilon_1, \epsilon_2, \ldots, \epsilon_m)$. The Jacobian matrix $G$ has an important block structure and is usually very sparse, usually containing only two nonzero blocks per block row. This is discussed in more details below. In solving (1), we are interested in the new velocities $v \in \mathbb{R}^n$, and the constraint forces $\lambda \in \mathbb{R}^m$. Vectors $p \in \mathbb{R}^n$ and $q \in \mathbb{R}^m$ are described below.

In case there are collisions or contacts between bodies, or any other form of nonsmooth dynamics, the stepping equations might require the solution of a complementarity problem, linear or otherwise. Even in that case, the local linear problems that need to be solved as the same form as given in (1).

The exact value of the vectors $p \in \mathbb{R}^n$ and $q \in \mathbb{R}^m$ is not directly relevant to the general discussion of iterative solution but in what follows, the following values are used

$$\begin{aligned} p &= Mv^{(0)} + hf^{(0)} \\ q &= -\frac{4}{h}\Upsilon g^{(0)} + \Upsilon G v^{(0)} + w^{(0)} \end{aligned} \tag{2}$$

where $v^{(0)} \in \mathbb{R}^n$ is the initial velocity, $f^{(\in)}\mathbb{R}^n$ are the forces acting on the system evaluated at the beginning of the time step, $h > 0$ is the time step, $g^{(0)} \in \mathbb{R}^m$ is the vector of constraint violation at the beginning of the time step, $\Upsilon$ is an $m \times m$ diagonal non-negative matrix, and $w^{(0)} \in \mathbb{R}^m$ is a given constraint velocity vectors that corresponds to nonholonomic constraints. Both holonomic

and nonholonomic constraints are treated together. For a nonholonomic constraint, the corresponding entry in diagonal matrix $\Upsilon$ vanishes.

The system matrix in (1) is not symmetric and thus, though it is positive definite, it cannot be processed using conjugate gradient or Gauss-Seidel [2] type of iterations. However, the Schur complement matrix

$$S_\epsilon = GM^{-1}G^T + \Sigma, \tag{3}$$

is symmetric and positive definite. The corresponding linear system becomes

$$S_\epsilon \lambda = q - GM^{-1}p. \tag{4}$$

The main stepping equation (1) can also be written as $v = v^{(0)} + hM^{-1}f^{(0)} + M^{-1}G^T\lambda$ which is interpreted as

$$\begin{aligned} v^{(1)} &= v^{(0)} + hM^{-1}f^{(0)} \\ f^{(c)} &= G^T\lambda \\ v &= v^{(1)} + M^{-1}f^{(c)} \end{aligned} \tag{5}$$

where $v^{(1)}$ is the predicted final velocity in case there are no constraints present, $f^{(c)} \in \mathbb{R}^n$ is the impulse (time integral of forces) due to constraints, and $v$ is the final velocity. A factor of $h$ was absorbed in $f^{(c)}$ for convenience. Using these definitions, the main linear system of equations to solve is

$$S_\epsilon \lambda = q - Gv^{(1)} \tag{6}$$

## 2 Gauss-Seidel iterations

To construct efficient G-S iterations on the linear system (6), we split the matrix $S_\epsilon$ as

$$S_\epsilon = L + D + L^T, \tag{7}$$

where matrix $D$ is block diagonal, symmetric and positive definite, and $L$ is strictly lower triangular. If the vector $\lambda$ is partitioned into blocks in the same way as matrix $D$, the iterative solution process at step $\nu > 0$, for the $k$th variable can be written as

$$D_{kk}\lambda_k^{(\nu+1)} + \sum_{j<k} L_{kj}\lambda_j^{(\nu+1)} + \sum_{j>k} L_{kj}^T\lambda_j^{(\nu)} = (q - Gv^{(1)})_k. \tag{8}$$

Adding and subtracting $D_{kk}\lambda_k^{(\nu)}$ from (8) produces the updating rule

$$D_{kk}\Delta\lambda_k = -\left(S_\epsilon\bar{\lambda} + Gv^{(1)} - q\right)_k, \tag{9}$$

where $\bar{\lambda}$ is the current value of $\lambda$, before the update, i.e., it contains all up-to-date known values of $\lambda$. Since it is common in Gauss-Seidel implementation to directly update all variables, $\bar{\lambda}$ is not labelled with the iteration $\nu$ but is understood to contain values from both the last and the current iteration sweeps.

Now, if the current constraint force is $\bar{f}^{(c)} = G^T\bar{\lambda}$, then, we have

$$S_\epsilon\bar{\lambda} = GM^{-1}\bar{f}^{(c)} + \Sigma\bar{\lambda}, \tag{10}$$

and therefore

$$q - Gv^{(1)} - S_\epsilon\bar{\lambda} = q - Gv^{(1)} - GM^{-1}\bar{f}^{(c)} - \Sigma\bar{\lambda}. \tag{11}$$

Now, note that what is sought is the computation of $v = v^{(1)} + M^{-1} f^{(c)}$ and thus, putting

$$\bar{v} = v^{(1)} + M^{-1} \bar{f}^{(c)} = v^{(1)} + M^{-1} G^T \bar{\lambda}, \tag{12}$$

the update equation can be written as

$$D_{kk} \Delta\lambda_k = -\left(G\bar{v} + \Sigma\bar{\lambda} - q\right)_k = -\bar{r}_k, \tag{13}$$

where the remainder $\bar{r}$ is defined as

$$\bar{r} = G v^{(1)} + S_\epsilon \bar{\lambda} - q = G\bar{v} + \Sigma\bar{\lambda} - q. \tag{14}$$

Now, once we have computed $\Delta\lambda_k$ as per (13), the update to $\bar{v}$ and $\bar{r}$ are as follows,

$$\begin{aligned} \Delta\bar{v} &= M^{-1} G_{\bullet k}^T \Delta\lambda_k \\ \Delta\bar{r} &= [S_\epsilon]_{\bullet k} \Delta\lambda_k, \end{aligned} \tag{15}$$

which means that for a given index or block $k$, a change in the Lagrange multiplier $\lambda_k$ will affect only the velocities of the bodies involved in the constraint corresponding to block or index $k$. The residual error, however, will be changed at all indices or blocks $c$ such that the constraint corresponding to index or block $c$ involves a body that is also involved in the constraint corresponding to index or block $k$. This last sentence is complicated on purpose: finding out how the residulal changes for a change in $\lambda_k$ is complicated and we're better off not tracing this complicated graph of interactions. Instead, we'll find a way to efficiently recompute the residual from scratch at each step below.

This algorithm can be implemented using only one temporary buffer, namely, the residual $r$, and writing directly onto the solution vector $v$ as described in Algorithm 2.1. Here, each index $k$ corresponds to a constraint which might either consist of a single equation or a block of equations of size $m_k$. Either way, for each constraint, there is a list of body indices, $b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$, and for each body $b$, there is a list of constraint $c_{b_1}, c_{b_2}, \ldots, c_{n_{c_b}}$.

**Algorithm 2.1** Gauss-Seidel iterations to solve $(GM^{-1}G^T + \Sigma)\lambda = q - Gv^{(1)}$

Given $q, M, G, \Sigma, hf^{(0)}$.
initialize $v = v^{(0)} + hM^{-1}f^{(0)}, \quad \lambda = \lambda^{(0)}, \quad r = Gv + \Sigma\lambda - q.$
Compute blocks $D_{kk} = \sum_b G_{kb} M_{bb}^{-1} G_{kb}^T + \Sigma_{kk}$, for $k = 1, 2, \ldots, n_c$
**repeat**
  **for** $k = 1, 2, \ldots, n_c$ **do**
    Compute $z = -D_{kk}^{-1} r$ ▷ Update residual
    $r = r + \Sigma_{kk} z$ ▷ Loop over bodies connected via constraint $k$
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do**
      $d_b = M_{bb}^{-1} G_{kb}^T z$ ▷ Loop over constraints connected to body $b$
      **for** $c = c_{b_1}, c_{b_2}, \ldots, c_{n_{c_b}}$ **do**
        $r_c = r_c + G_{cb} d_b$ ▷ Update residual
      **end for**
      $v_b = v_b + d_b$ ▷ Update velocities
    **end for**
  **end for**
**until** Error is small, or iteration time is exceeded

The main work done in Algorithm 2.1 the computation of the incremental force, $z = -D_{kk}^{-1} r_k$, and then, the update of the velocities $v$ and that of the residual vector $r$. Note that these last two updates are SCATTER types of operations where we have to find all bodies connected to a given constraint, and

then, find all constraints connected to the given bodies. Keeping these operations, most of the rest of the algorithm can be changed such as, for instance, the order in which the constraints $k$ are traversed, whether the full incremental force $d_b$ is used or some scaled version to introduce relaxation.

It is actually not necessary to update the residuals and skipping that term removes the need to keep a list of constraints acting on a given body. In turn, skipping this step makes it necessary to keep track for the multiplier $\bar{\lambda}$. Indeed, we have $r_k = G_{k\bullet}\bar{v} + \Sigma_{kk}\bar{\lambda}_k - q_k$, which can be written as

$$r_k = \sum_b G_{kb}\bar{v}_b + \Sigma_{kk}\bar{\lambda}_k - q_k, \tag{16}$$

which means that as long as there is a list of rigid bodies acted upon by a given constraint with index $k$, the residual can be computed locally. The full algorithm is listed in Algorithm 2.2. It is also possible to compute the residual $r$ for the initial configuration, and then, given the local residual computed before and after the velocity update, the global value can be updated.

**Algorithm 2.2** Gauss-Seidel iterations to solve $(GM^{-1}G^T + \Sigma)\lambda = q - Gv^{(1)}$, without residual update.

Given $q, M, G, \Sigma, hf^{(0)}$.
initialize $v = v^{(0)} + hM^{-1}f^{(0)}, \quad \lambda = \lambda^{(0)}$.
Compute initial residual and error, $r = Gv + \Sigma\lambda - q, \quad \xi = \|r\|^2$ ▷ Not strictly needed.
Compute blocks $D_{kk} = \sum_b G_{kb}M_{bb}^{-1}G_{kb}^T + \Sigma_{kk}$, for $k = 1, 2, \ldots, n_c$
**repeat**
  **for** $k = 1, 2, \ldots, n_c$ **do**
    $r = -q_k + \Sigma_{kk}\lambda_k$ ▷ Compute local residual from scratch
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do**
      $r = r + G_{kb}v_b$
    **end for**
    $\xi = \xi - \|r\|^2$ ▷ Not needed if error is not monitored
    $z = -D_{kk}^{-1}r_k$
    $\lambda_k = \lambda_k + z$
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do** ▷ Loop over bodies connected via constraint $k$
      $v_b = v_b + M_{bb}^{-1}G_{kb}^T z$ ▷ Update velocities
    **end for**
    $r = -q_k + \Sigma_{kk}\lambda_k$ ▷ Recompute local residual from scratch
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do**
      $r = r + G_{kb}v_b$
    **end for**
    $\xi = \xi + \|r\|^2$ ▷ Not needed if error is not monitored
  **end for**
**until** Error is small, or iteration time is exceeded

## 3 Complementarity conditions

In the case of a contact constraint where $g_k \geq 0$, there is a restriction on the constraint force so that $\lambda_k \geq 0$, as well as a complementarity condition between the constraint and the force so that $g_k\lambda_k = 0$. In order to enforce these conditions, we need to keep track of $\bar{\lambda}$ in addition to keeping track of the residual and the current estimate of the velocity. Consider for instance a rewrite of the main update equation so the unknown is now $z = \lambda_k^{(\nu+1)}$ instead of $\Delta\lambda_k$, so now, the main work to perform can be

written as

$$\begin{aligned} D_{kk}z + r_k - D_{kk}\lambda_k = 0, &( \text{ solve this for } z, \text{ respecting complementarity conditions}) \\ d_k \leftarrow z - \lambda_k, &( \text{ compute the differences } ) \\ \lambda_k \leftarrow z, &( \text{ update the forces } ) \ . \end{aligned} \tag{17}$$

Everything else being left unchanged, we can write the LCP version of this algorithm as follows

**Algorithm 3.1** Gauss-Seidel iterations to solve $(GM^{-1}G^T + \Sigma)\lambda = q - Gv^{(1)}$, without residual update.

Given $q, M, G, \Sigma, hf^{(0)}$.
initialize $v = v^{(0)} + hM^{-1}f^{(0)}, \quad \lambda = \lambda^{(0)}$.
Compute initial residual and error, $r = Gv + \Sigma\lambda - q$ ▷ Not strictly needed.
Compute blocks $D_{kk} = \sum_b G_{kb}M_{bb}^{-1}G_{kb}^T + \Sigma_{kk}$, for $k = 1, 2, \ldots, n_c$
**repeat**
  **for** $k = 1, 2, \ldots, n_c$ **do**
    $r = -q_k + \Sigma_{kk}\lambda_k$ ▷ Compute local residual from scratch
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do**
      $r = r + G_{kb}v_b$
    **end for**
    Solve LCP: $0 \leq D_{kk}z + r - D_{kk}\lambda_k \perp z \geq 0$
    $\Delta\lambda_k = z - \lambda_k$
    $\lambda_k = z$
    **for** $b = b_{k_1}, b_{k_2}, \ldots, b_{n_{b_k}}$ **do** ▷ Loop over bodies connected via constraint $k$
      $v_b = v_b + M_{bb}^{-1}G_{kb}^T\Delta\lambda_k$ ▷ Update velocities
    **end for**
  **end for**
**until** Error is small, or iteration time is exceeded

Of course, solving the LCP step can be problematic for contact problems but a reasonable solution was provided for this earlier in the course. One alternative and simple way to do this is to first solve:

$$z = -D_{kk}^{-1}r + \lambda_k, \tag{18}$$

by directly inverting the $3 \times 3$ contact matrix $D_{kk}$. Then, if the normal force $z_1$ is negative, set $z = 0$. Otherwise, if the tangential part is too large, i.e., of $z_2^2 + z_3^2 > \mu z_1^2$, just project the tangential vector as follows

$$\begin{aligned} t &\leftarrow \sqrt{z_2^2 + z_3^2}, \\ z_2 &\leftarrow (z_2/t)\mu z_1, \\ z_3 &\leftarrow (z_3/t)\mu z_1. \end{aligned} \tag{19}$$

This technique is free from singularity though it introduces some subtle anomalies as described by Anitescu [1].

One can also approximately solve the nonlinear complementarity problem as follows. First solve for $z$ and in Eqn. (18) and check that $t = \sqrt{z_2^2 + z_3^2} < \mu z_1$ (note that anisotropic case requries some subleties described below). If this is the case, you have a stiction solution that is consistent for that contact. Otherwise, you know that the solution to that frictional problem has the form $z = (z_1, z_2, z_3)$ where $(z_2, z_3) = -\mu z_1 u$, where $u \in \mathbb{R}^2$ is the unit vector in the direction of sliding. Assume now that this direction is $u = -w = (1/t)(\bar{z}_1, \bar{z}_2)$, where $\bar{z}$ is the solution obtained from Eqn. (18). Theoretically,

you would have to solve for $u$ so that the dissipation rate is maximized but let's skip that. We still have to compute $z_1$ for the correct solution, i.e., set $D = D_{kk}$ and $\lambda = \lambda_k$ for the moment, then

$$(Dz)_1 = -(r)_1 + (D\lambda)_1, \tag{20}$$

which reads

$$d_{11}z_1 + d_{12}z_2 + d_{13}z_3 = z_1(d_{11} + \mu d_{12}w_1 + \mu d_{13}w_2) = -r_1 + (d_{11}\lambda_1 + d_{12}\lambda_2 + d_{13}\lambda_3), \tag{21}$$

given our assumption about $z_2$ and $z_3$.

If one works with one single equation at a time, the LCP for the normal is just the scalar equation, i.e., $D_{kk} = d \in \mathbb{R}$ and so we just compute

$$z = -r/d + \lambda, \tag{22}$$

and project the result to the allowed region.

## References


[1] M. Anitescu, *Optimization-based simulation of nonsmooth rigid multibody dynamics*, Math. Program., 105 (2006), pp. 113–143.

[2] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, vol. 16 of SIAM Frontiers, SIAM Publ., Philadelphia, 1995.