Boundary Conditions

minchernl

February 2021

0.1 Boundary Conditions

Usually, boundary conditions (BC) in simulation are essentially linear equality constraints in optimization per time step

$$Bx = p, \tag{1}$$

where B is a $m \times dn$ matrix ($m \leq dn$) selecting the BC nodal DOFs, and p is a $m \times 1$ vector specifying the prescribed spatial values.

At the beginning of each time step, if x could be directly modified to ensure all BCs are satisfied, the constraint during optimization simplifies to

$$B\Delta x = 0, \tag{2}$$

where Δx is the search direction in each optimization iteration. This is because Equation 1 and 2 together ensure that the updated nodal DOF $x + \alpha\Delta x$ still satisfies the boundary conditions:

$$B(x + \alpha\Delta x) = p. \tag{3}$$

The two most common (Dirichlet) boundary conditions are sticky and slip:

Sticky boundary conditions are like fixing the position of certain nodes in a time step. It has a block-wise constraint Jacobian matrix B where each d rows contain exactly one $d \times d$ identity matrix and zero matrices elsewhere.

Slip boundary condition constrains each BC node inside a certain linear subspace, e.g. a plane or a line, which is not necessarily axis aligned. Taking planar slip boundary condition as an example, for each BC node there is one row in B that has the normal vector of the plane residing on the columns corresponding to the BC node. This constraint ensures that the displacement has zero component in the normal direction of the plane, so the node always stays inside the plane.

In what follows, we first introduce two methods (Section 0.1.1, 0.1.2) for handling equality constraints during each linear solve, taking sticky BCs as the example. The DOF elimination method (Section 0.1.1) is straightforward and efficient, but it does not apply to the situation where the DOF cannot be directly modified to satisfy the constraint at the beginning of a time step. In this situation, we need to apply the Augmented Lagrangian method (Section 0.1.2) to first "drag" the node to the prescribed location, and then switch to the DOF elimination method for faster convergence. Then in Section 0.1.3 we look into the transformation from slip BCs to the same form as sticky BCs so that they could be handled in the same spirit.

0.1.1 DOF Elimination Method

To handle linear constraints (Equation 2) in each Newton iteration while solving for the search direction Δx, we form the Lagrangian after quadratic approximation to the Incremental Potential

$$L(\Delta x, \lambda) = \frac{1}{2}\Delta x^T H \Delta x + g^T \Delta x + \lambda^T B \Delta x, \tag{4}$$

where λ is the $m \times 1$ Lagrange multiplier vector, and then derive and solve the KKT system

$$\begin{bmatrix} H & B^T \\ B & \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -g \\ 0 \end{bmatrix}. \tag{5}$$

However, this system is not SPD anymore, and its size also grows with the number of BC nodes.

Taking the simplest sticky BC as an example, since its constraint Jacobian B is a selection matrix, we know that BB^T is a $m \times m$ identity matrix, and B^TB is a $dn \times dn$ diagonal matrix with unit value entries only on those corresponds to BC nodes and zero entries elsewhere. Now if we left multiply B to the first block row of Equation 5, we can directly solve for

$$\lambda = -BH\Delta x - Bg. \tag{6}$$

Substituting Equation 6 back into the first block row of Equation 5, we obtain

$$(I - B^TB)H\Delta x = (I - B^TB)(-g). \tag{7}$$

Here left multiplying $(I - B^TB)$ is equivalent to zeroing out the rows corresponding to the BC nodes. Therefore in Equation 7 we get an under-constrained system with $dn - m$ equations but dn unkowns in Δx. But from the second block row of Equation 5 we in fact know the Δx at BC nodes, thus we can make the system back to SPD again by rewriting Equation 7 as

$$H_{UB}\Delta x_B + H_{UU}\Delta x_U = -g_U, \tag{8}$$

where

$$H = \begin{bmatrix} H_{BB} & H_{BU} \\ H_{UB} & H_{UU} \end{bmatrix}, \quad \Delta x = \begin{bmatrix} \Delta x_B \\ \Delta x_U \end{bmatrix}, \quad g = \begin{bmatrix} g_B \\ g_U \end{bmatrix} \tag{9}$$

and B indicates BC nodes without loss of generality on the possibly applied reordering. Now as we know $\Delta x_B = 0$, we end up with a SPD system that does not contain the BC nodes:

$$H_{UU}\Delta x_U = -g_U. \tag{10}$$

To avoid remapping DOFs, state-of-the-art simulators often directly modify the original linear system to solve for Equation 10. This can be done by setting all the entries on the rows of BC nodes in H and g to 0, and then for the columns of BC nodes in H, set all off-diagonal entries to 0 and diagonal entries to 1 or any other positive real number that maintains a good system condition. After solving this modified system, all solved Δx_U are directly in place and all Δx_B are 0.

Note that this DOF elimination method only works here when the sticky BC nodes are already in place at the beginning of the time step. If it is not true, the constraint

function in Equation 5 no longer has a zero rhs, and the DOF elimination method will not be applicable. This is not because that we could not solve for Δx in this way with constraints containing nonzero rhs, but rather the solved Δx may not be a descent direction. Intuitively, if the direction of Δx_B is pointing to the BC prescribed coordinates, it can possibly increase the Incremental Potential which is not aware of the BCs. But if Δx_B is simply 0 when the BCs are already satisfied, it is like minimizing the Incremental Potential with a subset of variables, which is still valid.

When the BCs cannot be satisfied by directly modifying x at the beginning of a time step possibly due to interpenetration or inversion safeguard, we can first apply Augmented Lagrangian method (Section 0.1.2) to "drag" the BC nodes to the target location, and then switch to DOF elimination method.

0.1.2 Augmented Lagrangian Method

0.1.3 General Linear Constraint and Slip BC

For more general linear constraints where B is an arbitrary $m \times dn$ matrix, applying Augmented Lagrangian follows the same way, but it requires some change of variable techniques for the DOF elimination method to be applicable.

The main issue is that if B is not diagonal mod reordering, it becomes impossible to apply the elimination tricks in Section 0.1.1. Therefore, we linearly map the DOF into another set of variables such that the constraints becomes trivial as before.

Performing a singular value decomposition (SVD) on the constraint Jacobian matrix B, we obtain

$$B = USV^T \tag{11}$$

where U and V are a $m \times m$ and $dn \times dn$ orthogonal matrix respectively, and S is a $m \times dn$ diagonal matrix.

Let $y = V^T \Delta x$, we can rewrite the KKT system (Equation 5) as

$$\begin{bmatrix} V^T H V & S^T \\ S & 0 \end{bmatrix} \begin{bmatrix} y \\ \lambda' \end{bmatrix} = \begin{bmatrix} -V^T g \\ 0 \end{bmatrix}, \tag{12}$$

where $\lambda' = U^T \lambda$ and the system has diagonal matrices at off-diagonal, which allows us to directly apply the DOF elimination method. After solving for y, Δx can be obtained by a matrix-vector product $\Delta x = Vy$. Although performing SVD on large size matrices is expensive, for constraints like slip boundary conditions that are usually decoupled per node, the SVD results in block-diagonal U and V which could be constructed procedurally in an efficient way.

Slip BC at node i can be expressed as

$$\mathbf{n}_i^T (\mathbf{x}_i - \mathbf{x}_i') = 0 \tag{13}$$

where $\mathbf{n}_i$ is the normal of the plane that node i is slipping, and $\mathbf{x}'_i$ is an arbitrary point on that plane. As illustrated before, if at the beginning of the time step x can be directly modified so that node i is on the plane, the constraint simplifies to

$$\mathbf{n}_i^T \Delta \mathbf{x}_i = 0. \tag{14}$$

Performing SVD on the row vector $\mathbf{n}_i^T$, we obtain

$$\mathbf{n}_i^T = \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^T = 1 \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{n}_i^T \\ \mathbf{m}_i^T \\ \mathbf{l}_i^T \end{bmatrix} \tag{15}$$

where unit vectors $\mathbf{n}_i$, $\mathbf{m}_i$, and $\mathbf{l}_i$ together form an orthonormal basis in 3D. Then it is clear that U is simply a $m \times m$ identity matrix, S is a $m \times dn$ matrix with nonzero entries (unit valued) only in those columns corresponding to the first DOF of each slip BC node per row, and V is a $dn \times dn$ block-diagonal matrix with the $d \times d$ basis blocks as in Equation 15 only on those corresponding to BC nodes and $d \times d$ identity matrix elsewhere:

$$U = \begin{bmatrix} \ddots & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}, \; S = \begin{bmatrix} \ddots & & & & & & & \\ & 1 & 0 & 0 & & & & \\ & & & & 1 & 0 & 0 & \\ & & & & & & & \ddots \end{bmatrix}, \; V^T = \begin{bmatrix} \ddots & & & & \\ & \mathbf{n}_i^T & & & \\ & \mathbf{m}_i^T & & & \\ & \mathbf{l}_i^T & & & \\ & & \mathbf{n}_{i+1}^T & & \\ & & \mathbf{m}_{i+1}^T & & \\ & & \mathbf{l}_{i+1}^T & & \\ & & & & \ddots \end{bmatrix} \tag{16}$$

To compute $\mathbf{m}_i$ and $\mathbf{l}_i$ from $\mathbf{n}_i$, we first note that there are an infinite number of possible solutions. So, we can simply first construct $\mathbf{m}_i = \mathbf{n}_i \times \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ or $\mathbf{m}_i = \mathbf{n}_i \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ if $\mathbf{n}_i$ is almost colinear with $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, and then construct $\mathbf{l}_i = \mathbf{n}_i \times \mathbf{m}_i$. To obtain $V^T(-g)$, one only needs to left multiply each $\mathbf{V}_i^T = \begin{bmatrix} \mathbf{n}_i^T \\ \mathbf{m}_i^T \\ \mathbf{l}_i^T \end{bmatrix}$ to $-\mathbf{g}_i$. As for $V^T H V$, first left multiply each $\mathbf{V}_i^T$ to every block on block row i of H to obtain $V^T H$. Then for each block column i of $V^T H$, left multiply $\mathbf{V}_i = \begin{bmatrix} \mathbf{n}_i & \mathbf{m}_i & \mathbf{l}_i \end{bmatrix}$ onto every block. Finally, after solving for y by applying the DOF elimination method on the modified system (Equation 12), Δx can be obtained by $\Delta x = Vy$ with similar block(node)-wise operations.

For slip boundary condition on a line $\mathbf{n}_i$, simply calculate $\mathbf{m}_i$ and $\mathbf{l}_i$ and form $\mathbf{V}_i$ and then V in the same way as above. But note that now $\mathbf{U}_i$ becomes a 2×2 identity matrix, and $\mathbf{S}_i$ becomes $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.