# Proposed handling of friction in Simbody

Michael Sherman, 28 Dec 2010

## Background

Simbody has been used so far with compliant contact models, where the normal forces are conveniently available as functions of state. These are used with a continuous friction/stiction model that means that Coulomb friction forces in those contacts can also be state based. These models provide rich functionality and have many uses but they also have some drawbacks, for example:

* They don’t work for friction in internal-coordinate joints, such as a friction torque in a pin joint that is proportional to the radial reaction force.
* They can create stiff equations of motion that require implicit integration for speed.
* They require elaborate material properties and produce detailed information that may not be necessary for some problems of interest.
* There are problems, such as operational space control in robotics, where an instantaneously linear contact model is required or preferable.

For internal coordinate joint friction, the reaction force is acceleration-dependent and the friction force has the character of a constraint rather than a simple force – the friction force may in turn influence the reaction force as well. Similarly, any friction forces that arise within a constraint element will depend on the constraint forces which are again acceleration dependent.

I would like to achieve two improvements to Simbody:

1. Provide for reaction- and constraint-force based Coulomb friction, and
2. add unilateral constraint-based, coefficient-of-restitution rigid contact and friction as a user-selectable alternative to the compliant models.

## Current formulation

Simbody’s equations of motion currently incorporate constraints that fit the following model at the acceleration level:



or 

where *M*(*q*) and *G*(*q*) are the usual mass and constraint matrices. This allows for nonlinear holonomic and nonholonomic constraints, and linear acceleration-level constraints.

In forward dynamics, Simbody uses your spatial algebra algorithms for calculating the accelerations, and gets constraint multipliers by solving



Where is a possibly-singular symmetric matrix, and. If some of the constraints are unilateral, switching and complementarity conditions may be added and equation could be solved as a linear complementarity problem to determine both the active set of constraints and their multipliers. The same equations can be used to calculate impulses to respond to rigid collisions.

In the current formulation, the constraint force transmission matrix *GT* is just the transpose of the constraint Jacobian *G*. But that is only true for “non-working” constraints. (Q: prescribed motion constraints do work but also use *GT* – what is the right way to say this?) Stiction constraints, such as occur due to friction that induces rolling contact, fit the above model, since their dependence on normal forces appears only in switching conditions. Stiction constraints are thus ordinary nonholonomic constraints forcing relative velocities to be zero; there will be two constraint equations associated with stiction in a plane, with switching condition  used to decide when to inactivate a stiction constraint and switch to sliding (Coulomb) friction.

## Proposed formulation for sliding friction

Sliding friction does not fit the above model. In this case the friction force magnitude  depends on some normal force that may be a reaction force in a joint or a constraint force, that is,  or . Then the applied friction force will be given by  for some  that provides the direction of application and the mapping to generalized coordinates. (I called this *W* to suggest that it is the force transmission matrix for “working” constraints.) The direction of application of friction depends on sliding velocity, so we have *W*=*W*(*q*,*u*)[[1]](#footnote-1).

Here is the augmented system I’m contemplating for Simbody, introducing one scalar multiplier  for each sliding friction constraint:



or 

Here matrix *R*(*q,u*) gives the “reaction force” dependence of a particular friction force while *C*(*q,u*) gives its “constraint force” dependence, and *D*(*q,u*) reflects scaling of and interdependence among . (Q: is there a more elegant alternative or better notation? Should *D* just be an identity matrix and *bw*=0?)

Defining , , , , and we can rewrite equation as



Equation is similar to the maximal-coordinate formulation in Mitiguy and Banerjee[[2]](#endnote-1) but here we include friction dependent on internal coordinate joint reactions, while they needed only to deal with constraint reactions.

Because M is invertible, we can solve this for the multipliers as in equation :



where . Equation looks just like equation but with the symmetry of lost in the still-square matrix . Computationally, I don’t think this matters much since (a) we expect to be small in an internal coordinate system, (b) even the original was likely to be singular, and (c) both equations need to be solved with an LCP solver in the presence of unilateral constraints.

If Simbody can use equation to handle friction along with the rest of the constraints, it would be convenient for moving to general unilateral constraints. In that case the only change is that complementarity conditions would be added and solved along with equation with an LCP method that would solve simultaneously for the set of currently active constraints and their multipliers, or for currently impacting constraints and their impulses.

Because reaction-dependent friction is included in the same way as constraint-dependent friction, the method described by Baraff[[3]](#endnote-2) can be used during execution of the LCP to deal with indeterminacy and inconsistency, such as when a friction “lock up” has occurred that requires a tangential impulse to arrest the sliding velocity and initiate sticking.

# Formulation with prescribed motion

Here there are three categories of constraint on system accelerations: (1) prescribed motion in which a generalized acceleration is known a priori as a function of time, configuration, and velocity; (2) ordinary non-working (“ideal”) constraints; and (3) working (“non-ideal”) constraints, usually friction. All constraints are assumed to be linear in the generalized accelerations.

We’ll distinguish three kinds of constraint multipliers: τ for prescribed motion (in generalized force units), λN for non-working constraints, and λW for working (friction) constraints.[[4]](#footnote-2)









Substituting from eqn. , combining and , and moving known terms to the right gives





Now let

   

  

  









We want to solve this system efficiently using available operators. To do that, we’ll first solve for  (and simultaneously for the set of active constraints), then  and together via a hybrid forward/inverse dynamics recursion as described in Jain[[5]](#endnote-3).

Putting and together in matrix form:



Since , the “forward dynamics” part of the mass matrix, is invertible we can use eqn. to eliminate  from eqn. leaving us with  equations for the  unknowns :



We solve eqn. using a method that can deal with the facts that (a) the matrix on the left is likely to be singular (due to redundant constraints, for example), and (b) the constraints may be unilateral in which case we must solve simultaneously for the set of active constraints and their multipliers. 

# Practical description of constraints

Simbody supports generalized constraints and users must be able to write their own in a reasonably straightforward manner. Every constraint object supplies at least two methods: (1) a method that calculates constraint errors at the acceleration level as a linear function of acceleration and multipliers, and (2) a method that calculates constraint forces as a linear function of multipliers. For constraints that also act at the velocity or position level, methods that calculate velocity and position constraint errors must also be specified.





Unilateral constraints must satisfy additional conditions:







# Planar friction

This is the hardest case we encounter in practice. For rigid contact we address only a point contact between bodies A and B, and assume we are able to define a point in space O at which the contact is said to occur, and a normal direction **n**. Define a ground-fixed contact frame C with origin O, and axis **z**=**n**. Then the arbitrary **x** and **y** axes span the tangential (slip) plane. Define stations PA fixed to A and PB fixed to B that are instantaneously coincident with O. Define separation **p**=PB-PA , relative velocity , and relative acceleration(these are material point derivatives because PA and PB are treated as fixed on their bodies).

For friction calculation, we assume that the normal contact is active (****) and generating a normal force at PB and  at PA. *N* may depend linearly on constraint forces or mobilizer reaction forces, and may be signed. We want to determine a tangential friction force  such that  for some slip velocity-dependent friction coefficient with . (is applied at PB, at PA).

There are two distinct cases, depending on whether there is sliding at the contact point.

1. When relative tangential velocity, we are sliding. The friction force is calculated using a single-equation, working, acceleration-only constraint, and generates force  where  is the direction of tangential relative velocity (a unit vector).
2. When  (or close enough) there are two possibilities:
3. We are sticking: a stiction (rolling) constraint  (two equations, non-working, non-holonomic) is active, and able to prevent sliding by generating a constraint force  such that , or
4. We are about to start sliding: a transition constraint (one equation, working, acceleration-only) is active generating  where  is the direction of tangential relative acceleration (a unit vector).

Case 2b above is problematic, because  is nonlinear in the acceleration-level variables (both *N* and **a** are acceleration-dependent).

Here the switching conditions are nonlinear because they involve the magnitudes of the planar tangential force and acceleration vectors.

,





where , .

When constraint is active, . If , we’re sticking. If , we’re in transition and the rolling constraint should be replaced with the working constraints ,, where . Oops – nonlinear in  or . To make this linear, we replace with  where  are the multipliers just calculated by assuming the rolling constraint was enabled, that resulted in  so we’re going to produce a force in the same direction (but note that the normal force *N* may now change since it can depend on ). Note that this is only necessary when the stiction constraint is a candidate, meaning the slip velocity *v*=0. As soon as there is any slip velocity vector *v*, we switch to a single working constraint  with applied force , where  will drop smoothly from  to  over a small velocity range.

]

Our solver for calculating using eqn. must ensure that conditions - are satisfied by the solution, by partitioning the constraint equations into active () or inactive ().

# Algorithm

We want to find the minimum 2-norm solution  to the underdetermined, mixed, slightly nonlinear complementarity problem (NCP):



The nonlinearities are quadratic dependencies among the unknowns that arise from vector directions involved in friction calculations. The first step to solving the above system is to linearize it by finding current directions *d* and then holding them constant in the problem statement. Each linearization comes with a measure of the angular error in the frozen direction; when that exceeds a threshold we must relinearize before continuing. The goal is to solve this as a series of LCPs rather than as an NCP.



Linear problem: 



LCP: 



NCP: 

Equations and are used together to solve the linear system for  given just the reduced set of constraints selected by *s* (that is, rows of *A* and *b* and columns of *W* have been removed). Then equation is used with a complete, 0-paddedto calculate the rest of the acceleration errors for use in equation to solve for the unknowns *s* in equation by determining which of the eligible unilateral constraints are selected. Equation is used to measure the angular error between the directions used in the linearized model vs. those in the nonlinear model with the same active constraints. If any direction is too far off we relinearize by calculating new values for *d* and then restart the LCP on the new linear subproblem with the same selected set.

## Selection

Let *s* be a boolean selection vector, with *si*=1 for a selected constraint, 0 for not selected. Selection matrix *S* is an identity matrix whose *i*th row is removed if *si*=0, so left-multiplying by *S* keeps just the selected rows and right-multiplying by *ST* keeps just the selected columns.is constructed the same way but using the complement to select inactive rows. The goal of our solver is to determine *s*, , and  (all the other are zero). Rewriting and to use the selection matrix would give:







Then when we have selected the correct set of constraints, and linearization error  is in range,  is the solution.

1. Given *s*, solve for a least squares set of multipliers , using . Set .
2. Calculate , then , and .
3. If  (that is, there is a residual aerr for an active constraint), then we failed to solve the linear system. Some of the constraints are inconsistent. This may be a lockup configuration, or it may require an impulsive velocity change (friction jamming). TODO: how to tell?
4. If all inactive  and all active , we’re done and  is the solution.
5. If any inactive , those are redundant constraints that should be active so that they can carry some of the load. Activate them and return to step 1.
6. Otherwise select the “most violated” constraint and swap its active status. If this produces a value of *s* we’ve seen before, we have failed to find a solution and must give up in despair.
7. Otherwise, return to step 1.

What is the “most violated” constraint? We’re looking for the violated constraint *i* for which the change  yields the largest change in . For active constraints, that is just the most negative . For inactive constraints, it is the constraint *i* whose negative  would require the largest  to drive to zero.

# Example: point on line constraint

Here we always have two bilateral holonomic constraints *px*=*py*=0 that produce a constraint force that make point P follow a line in the **z** direction.

* In stiction, we also have non-holonomic constraint *vz*=0 and we enforce *az*=0 with force  provided that . To write the *vz*=0 case as a complementarity problem, use four constraints per Pfeiffer and Glocker. Write , .





Then . Note that eqn. is nonlinear in  because .

* In transition (included above in eqn. ) we have  which is split into two complementary constraints that are quadratic in .
* During sliding, we have  and working constraint



where again . Equation contains the same nonlinearity as eqn. .

#### Linearizing

There are two nonlinearities to deal with. One is introduced by during transition and is dealt with by splitting the stiction/transition problem into “left” and “right” halves, with only one of those selected at a time. The other nonlinearity is introduced by the normal force magnitude *N*, which is quadratic in  both in transition and sliding.

We linearize the magnitude calculation by calculating , saving the direction  and then defining on later evaluations. Then the angular error in the linearized system is given by  .

#### LCP

To write the *vz*=0 case as a linear complementarity problem, use four constraints per Pfeiffer and Glocker. Write , . Define . The constraints are



with complementarity conditions





# Example: unilateral point in plane constraint

Here we have a unilateral holonomic constraint that produce a constraint force that makes point P move only in the **x**-**y** plane (i.e.,  during contact).

* In stiction, we also have two non-holonomic constraint equations  and we enforce with force  provided that , that is,  ( is linear in the unknowns but the limiting condition is quadratic).
* If the force would be too big (transition to sliding), then we set which is nonlinear in .
* During slipping we have instead  which is linear in .

#### Linearizing

We linearize by selecting the stiction constraint and calculating , saving the direction  and then using that direction to approximate tangential force magnitude during stiction and as the force direction during transition.

* Stiction limiting condition is . Linearization error is .
* Transition force is . This should oppose the resulting acceleration direction so we require. Linearization error is .
* Sliding force  is already linear.

#### LCP

For the stiction-eligible case ():



# Example: friction in a pin joint

Here we have available the spatial reaction force *F* (meaning force **f** and moment **m**) at a pin mobilizer with generalized speed . Reaction  is linear in the generalized accelerations , but may in general involve all of them, not just  (and  is linear in constraint multipliers ). We’ll assume here that Coulomb friction is to be generated from the magnitude of the radial component of reaction force, ignoring reaction moment. We decompose reaction force and define normal force magnitude  which is quadratic in the unknowns.

* In stiction () we introduce a single non-holonomic constraint equation  and enforce  with constraint force that is a generalized force (a torque in this case) acting at mobility , provided (the friction force is linear but the limiting condition is quadratic).
* If the force would be too big (transition to sliding), then which is nonlinear (can be split into two complementary quadratics) in the unknowns.
* Finally, in sliding () we have  which is quadratic.

# Alternate method without working constraints

The idea here is to treat *sliding* friction forces as applied forces rather than as constraints. This moves the nonlinearities into the force calculations rather than the constraints, with the exception of the stiction switching condition which will still be quadratic due to the need to calculate vector magnitudes. Here is the alternate formulation:









We have a forward dynamics operator *dyn*() that can efficiently calculate three of the unknowns given an estimate for the friction multiplier:



From this we can estimate the error in the friction multiplier:



A fixed-point iteration is constructed by setting  repeatedly until  goes below some prespecified tolerance. A Newton iteration can be constructed as follows:



We start these iterations with normal force  saved from the end of the previous step. The factored Jacobian can also be saved and reused in equation for multiple steps until convergence is too slow. Failure to converge with a fresh  most likely represents an inconsistent state (see Painlevé’s paradox) and requires an impulsive velocity change.

Once  has been calculated to sufficient accuracy, the set of active constraints and sliding friction forces must be checked for consistency. If any of the unilateral constraint conditions are violated, a new active set must be chosen and the process repeated.

For each friction element *k*, there is an associated slip velocity  and slip acceleration . During sliding, we know the force direction () so friction force element *k* is driven by a nonnegative scalar  and  to generate a generalized force or pair of spatial forces opposing the sliding velocity. Because  may depend quadratically on the unknowns, calculation of  is nonlinear (though only quadratic).

This event witness function detects transitions from sliding to sticking: . A sign change from positive to nonpositive means we might be sticking; the time stepper will ensure that the zero crossing is isolated to within a small time window  so *e* will be very close to zero when the event handler is called. [TODO: should be able to guarantee that .] At this point we must determine whether a friction force of magnitude  can enforce ; if so we’re in stiction. We make the non-holonomic constraint  eligible and solve the LCP using its derivative  under the (quadratic) condition . If this inequality holds, the constraint is active and we’re in stiction. [TODO: If we can’t guarantee that  already we’ll have to project here.]

During the LCP, if , the constraint is inactivated and is replaced instead by a transition force and then iterated until , stopping when the two vectors are the same to within a relative tolerance (fraction) *ftol*; that is, the magnitude of their difference is . We expect neither *N* nor **a** to become zero during the iteration, and failure to converge means that  was sufficient to reverse the acceleration, meaning that a constraint force below the limit should have been able to stop it.

During transition, the friction magnitude is limited to and the unknown is a vector quantity in Cartesian or mobility space so each transitioning friction force is driven by an unknown vector  where  is the direction of impending sliding. In this case  will have only 1 and -1 in the populated elements.



# Compliant contact with stiction constraints

This is a much simpler case that the ones above, because for a compliant contact the normal force vector  is state-dependent, that is, . Stiction constraints act at the acceleration stage so can’t affect . Every active contact is in one of three “friction states” we’ll call *Slip*, *Creep*, and *Stick*, with the current state maintained as a discrete state variable. Changing the value of the friction state invalidates the Dynamics stage of the whole State since it affects what forces are applied.

While in one of these states, the system is continuous in time but watches for event triggers that can cause state transitions.

## State “Slip”

A contact is in this state when the contacting bodies are slipping “quickly” against each other so are experiencing dynamic friction. More precisely, where  is the tangent-plane slip velocity and *vtol* is the slow speed below which we are willing to consider that the bodies might be sticking together. During sliding, the friction force magnitude is and its direction is , a perfectly ordinary velocity-dependent force. The friction coefficient may be velocity dependent at low velocities, with . We monitor an event trigger function  and invoke the Slowdown Handler if we see a falling transition  indicating a slowdown or reversal of slip velocity. Hereis the previous slip direction, which we update at every step. (If  we use  as the direction instead.) This allows us to spot slip reversals, which could be missed if we were to use as a trigger function.

## State “Stick”

A contact is in this state when it has previously been determined that the contacting bodies are in stiction, that is, their tangent-plane slip velocity is zero and the friction force **λ** necessary to keep it there doesn’t exceed . A no-slip constraint  is active and provides the necessary friction force as a constraint reaction during realize(Acceleration) instead of the force element that is used while slipping. This is a planar constraint so there are two constraint equations and . Since the velocity is forced to be zero, we can only leave the stick state if the constraint force exceeds the limit. (You can’t leave the Stuck state because of constraint drift!) So we monitor a trigger function  and invoke the Breakaway Handler if we see a falling transition , indicating that the force got too big. This is actually an oversimplification because it doesn’t deal with the case where the constraint equations are unable to satisfy the constraint; we’ll add that refinement below.

Note that the applied constraint reaction  may exceed the limit for the stiction force; that is not a problem for realize(). If the result isn’t consistent, it is up to the integrator to notice that the transition has occurred and do something about it. I don’t think that’s right since might get too big during the “LCP-ish” solve, leaving it eligible but not active.

## State “Creep”

This state is most commonly entered via the Breakaway Handler that is invoked when a formerly-stuck contact comes unstuck due to excessive stiction forces. Here the slip velocity is below tolerance, but we cannot enforce stiction. Creep can also be entered via the Slowdown Handler if the stiction constraint can’t be used. We will likely be in this state only briefly. Most commonly, the slip velocity will increase and we will transition to Slip. However, it is also possible that the contact will transition (back) to stiction. So we watch for two event triggers, using the functions introduced above but with opposite sign transition directions:  and . Trigger function  is realized at Velocity stage. At Dynamics stage we apply a friction force of magnitude  opposing the slip velocity if it is large enough, otherwise opposing the last-known “impending slip” direction. At Acceleration stage, we enable the  acceleration constraint, and calculate what the corresponding **λ** would have been if we hadn’t applied a friction force. Then we can evaluate  to determine whether we should switch to the Stuck state. We also update the “impending slip” direction  to use next time if the velocity is too small.

## What about redundancy/poor conditioning?

Despite our best intentions, it is possible that the stiction constraint cannot enforce the no-slip condition. For example, the stiction constraint may conflict with some other constraint or prescribed motion. In that case the constraint might be removed, resulting in , or it may get a “least squares error” value which is non-zero but insufficient. For that reason we must look at the no-slip acceleration-level constraint to verify that it has been satisfied, prior to checking whether the magnitude of is acceptable. We can embed that in the  function by setting it to  whenever either of the two constraint equations is not satisfied with the constraint enabled.

Here is a simple contrived example that illustrates that stiction constraints can’t be treated in isolation:

N1

N2

F=1.5µN

λ1

λ2

This is a 1-dof cart fixed to a frictionless rail and sliding on soft pads (not rolling wheels) against a hard floor. The pads have deformed to yield the contact points shown with red dots, and the deformation is generating the normal forces shown (the equal and opposite force acting on the floor isn’t shown). Assume for a moment that N1=N2=N and µ1=µ2=µ. We apply a force F to the cart with magnitude 1.5µN. Enabling one constraint equation would reduce this system from 1 to 0 dofs; therefore the two available stiction constraints must be redundant. Yet each friction force is limited to µN, so neither friction constraint alone can stop the cart. But with both constraints enabled, a least squares solution for λ1 and λ2 would result in each contributing a force of 0.75µN, easily in range and stopping the cart. If the constraints are scaled by their limit forces µ*i*N*i*, this can be used to produce different λ*i*. This makes the constrained stiction solution very close to the result of applying the expensive continuous friction model to this problem – it would generate friction forces on both pads that together would stop the cart (except for the stiction drift that is part of that model).

Slowdown handler

Lockup handler

Slip handler



Breakaway handler















Each active contact is in one of three states, during which the simulation is continuous but with one or both of the trigger functions eslip and estick active. When the indicated sign transition occurs (red arrows), a handler is called. The slip and breakaway handlers just set the state. The lockup handler sets the state, removes residual velocity with a momentum-balanced impulse, and enables the no-slip constraint. The slowdown handler evaluates estick and then dispatches to the lockup handler or breakaway handler as appropriate.

# Formulation with reaction-dependent forces, take 2

Here is a high-level view of the equations of motion suitable for the discussion here:



The unknowns are: the subset *s* of eligible constraints that are active, the unprescribed generalized accelerations , the constraint multipliers *λ*, the prescribed motion forces *τ*, and the friction magnitudes *γ*. Forces, and the prescribed generalized accelerations  are functions of state.  consists of problematic applied forces that may depend on the values of the unknowns. In practice these are almost always friction forces dependent on either mobilizer reaction forces or constraint reaction forces; for compliant contact they are state-dependent but present only if the corresponding stiction constraint is not active. Friction dependencies when present are linear or quadratic in the unknowns.

We’ll distinguish three kinds of constraint multipliers: *τ* for prescribed motion (in generalized force units), *λb* for unconditional (bilateral) constraints, and *λc* for conditional constraints. The other unknowns are  (the “regular”, non-prescribed generalized accelerations), friction magnitudes *γ*, and the selection variables *s*. Note that *s* selects all or none of the constraint equations associated with a Constraint.



B is the ordinary unconditional constraint Jacobian; C is the constraint Jacobian for all kinematically eligible conditional constraints. The above is how the system looks if all the eligible constraints are active; below we’ll include the more realistic scenario in which only a subset are active.

We’re not allowing for friction forces that depend on the prescribed motion forces *τ*; that could be allowed but seems unnecessary.

Depending on the form of the selection criteria for *s*, it may be possible to solve simultaneously for *s* and *λ* using, for example, an LCP. Friction introduces nonlinearities that make that difficult; the standard treatment adds working constraints and linear approximations to deal with friction; we want to keep the constraints non-working, allow for an exact solution, handle redundant constraints properly, and support friction that is dependent on mobilizer reactions as well as constraint reactions. An iterative approach is more appealing for that: choose s; iterate to obtain , , and *λ*; update s.

In more detail, an iterative solution procedure looks like this:

1. Assume an event handler has determined the set of mc conditional constraints that are kinematically eligible to be active. The sizes of *C*, *λc*, and *s* have been adjusted. We have to determine the subset *s* of eligible constraints that are actually active, and the corresponding values of the unknowns *λ*,, *τ* and *γ*.
2. Operators M, G, and W are available from the state t,q,u,*s*prev.
3. Calculate from the state.
4. Initialize elements of *s* to their previous values or 1 for newly-added eligible constraints.
5. Guess values for ffriction based on previous values of *s* and *N* with .
6. Calculate . (Don’t iterate ffriction yet; it does not depend on .)
7. Calculate *λ* and constraint reaction forces and revise *s* until consistent (LCP-ish). Calculateand prescribed motion forces *τ* from the complete set of forces and determine the mobilizer reaction forces.
8. Now update ffriction, check convergence, and return to step 6 until converged.
9. Check constraint conditions. If any are violated, update *s* and return to step 4.

To get a good look at the solution procedure, we’re going to split the equations into prescribed and “regular”. To simplify notation, we’ll abbreviate some symbols and use a hat to mark symbols that have had rows and/or columns removed to include only active constraints.



Then











Splitting the “p” rows from the “r” rows and moving known terms to the right gives







Then using the fact that *Mr* is invertible and combining with gives us an equation we can solve for the multipliers (*λb*,*λc*) if we hold the friction forces at an estimated value *γ*.



where 

and .

We obtain a least squares solution for *λ* using a pseudo inverse:



Then from equation 

and from equation 

The last two equations are solved together by our recursive algorithm. Eqn. gives us from . The orange shows how the friction forces affect the unknowns; by expanding you can see the explicit linear dependence of the other unknowns on *γ* (this is just FYI):







Note that friction forces affecting prescribed mobilities are simply canceled by the prescribed motion torques; otherwise they have no effect.

## Solving for friction forces

Our friction force elements are used during creep and slipping phases of a frictional contact, instead of the no-slip constraint used during stiction. The selection variable *s* determines whether the constraint is active or whether the corresponding friction force element is active.

Each friction force element is driven by a single unknown scalar *γ* representing the *µN* magnitude of the friction force acting to oppose a sliding direction **v**. *γ* can thus be quadratically dependent on a subset (possibly large) of the and *λ* unknowns, through the reaction forces *N* they generate, although for compliant contact *N* is known as a function of state.

Given  for the *k*th friction element, the friction forces (body or mobility) for that element are linear in :



, *λ* and *τ* in turn depend linearly on *w* as shown in equations -.

*D* contains the force application direction –**v**, where **v** is either the velocity-dependent slip direction, if slip is large enough, or a pre-recorded “impending slip” direction based on the direction of the (excessive) constraint force generated when we last tried to use a no-slip constraint for stiction. Either way we do not allow *D* to change while we are solving for , *λ*, *τ* and *γ*.

### Determining the impending slip direction

When the stiction constraint is eligible (meaning slip velocity below *vtol* so we’re in creep phase) we have to activate it, calculate the normal force *N* and the friction force **λ** it would generate, and use it if it is able to satisfy the stiction constraint with . Otherwise, we record one of two directions. If the constraint was satisfied but the force was too big, then. Otherwise, we were accelerating and we record . Then **v** is the impending slip direction. Then we disable the stiction constraint, update *D* from **v**, and evaluate the friction force instead using *γ*=*µN* as the initial guess. Note that this is part of the computation of the set *s* of active constraints.

### Iteration

We start by initializing , where typically we will have estimated  using a normal force that we saved from the previous step. Note that this doesn’t necessarily give us since  can be a function of state and may have changed. Now we can calculate  and evaluate the error



In matrix form:



where is a diagonal matrix with the  on the diagonal and *N* is a column vector of all the normal force magnitudes. A Newton iteration would then use the following update scheme:



where 

This Newton iteration would converge very fast since *N* is linear or quadratic in *γ*. Also, diagonals of  will typically be zero or nearly so[[6]](#footnote-3), although some of the off-diagonals might be significant. Nevertheless, there is hope that ignoring the second term in eqn. will still yield a rapidly convergent fixed-point iteration, using. If we do need to use , it can be expected to change slowly so a numerical approximation might work for many steps. And if uncoupled contacts can be identified and handled in small groups, the numerical approximation and small matrix inverses could be done efficiently.

With whatever method, we iterate until  for some suitable scaled norm and relative friction tolerance *ftol*. This procedure results in a consistent set of values for the unknowns given the set of active constraints *s* and the impending slip directions **v** that are in some of the friction force transmission matrices *W*.

1. Just at the transition from sticking to sliding, the direction cannot be determined from the slip velocity. In that case it is customary to assume that the initial sliding force has the same direction as the final sticking force. [↑](#footnote-ref-1)
2. Mitiguy, P.C. and Banerjee, A.K. Efficient simulation of motions involving Coulomb friction. *J. Guidance, Control, and Dynamics* 22(1) 1999. [↑](#endnote-ref-1)
3. Baraff, D. Fast contact force computation for nonpenetrating rigid bodies. *ACM* *SIGGRAPH Proceedings*, 1994. [↑](#endnote-ref-2)
4. For efficiency, we don’t permit the working constraints to depend directly on the prescribed motion forces *τ*, although they can depend on the resulting accelerations. This is unlikely to be useful in practice, but if needed it can be done by including a prescribed motion constraint in equation , meaning one of the λN multipliers will be the prescribed motion force, and a working constraint could depend directly on that. [↑](#footnote-ref-2)
5. Jain, A.; Rodriguez, G. Recursive dynamics algorithm for multibody systems with prescribed motion. *J. Guidance, Control, and Dynamics* 16(5):830-837 (1993). [↑](#endnote-ref-3)
6. That’s because friction is perpendicular to normal force. For friction to affect the normal force it depends on requires coupling through the mechanism, certainly possible but likely to diminish the effect substantially. [↑](#footnote-ref-3)