1. Ran 2D-RZ_finiteStrain_test.i for more complex meshes and longer times
    - changed dt in test file (too large for model beyond one time step)
    - changed PressureTM BC (from factor to function)
    - also edited 3D-RZ_finiteStrain_test.i
    - marked 3D (fake/axisymmetric 3D) test 'heavy' to be skipped on run_tests
    - re-golded, merged on github (issue #5584)
2. Created master SM vs TM list
    - all categories (bcs, materials, auxkernels, etc.)
    - missing items, replaced items, etc.
3. Added save_in_disp_r/z
    - aux variables to save r/z displacement residuals
    - file changed: TensorMechanicsAxisymmetricRZAction.C
        - to match TensorMechanicsAction.C
    - added test: tensor_mechanics/tests/2D_geometries/2D-RZ_finiteStrain_resid.i & gold
    - merged on github (issue #5610)
4. Compared timing of TM J2 tests vs LSH SM tests
    - added j2_plasticity_vs_LSH directory in modules/combined/tests
        - side-by-side comparison of TM vs SM for the same problem (no hardening)
        - time discrepancy for plastic steps (TM 5-7x slower with 4x4x4 mesh) (Figure 1)
    - merged on github (issue #5628)
5. Worked on TM timing by adding special J2 case
    - wrote radial return map and radial consistent tangent operator for J2 plasticity
    - worked with Andy to move radial return map to J2 userObject and set up for future custom return maps & consistent tangent operators for other userObjects
    - timing vastly improved (TM now less than 2x slower)
        - previously: 5.57s SM vs 34.21s TM (ten time steps)
        - now: 3.49s SM vs 5.75s TM (21.28s SM vs 35.56s TM for 50 time steps)
    - created j2_hard2_exp.i test for J2 with exponential hardening
        - plotted vm stress vs intnl to check exponential relationship, correct (Figure 2)
        - plotted same for j2_hard1 (constant/no hardening), correct (Figure 3)
    - waiting for Andy then PR & merge
        - still needs to be re-golded, j2_hard2_exp.i to be added to test (csvdiff?)
    - pushed on github (issue #5628 & #5667), branch returnMappingTM_5667
8. Coded multi surface plasticity, n internal parameters per model
    - started work on this but did not finish
        - most coding done, getting segfault / out-of-bounds
        - pushed on github, branch CMPS_multi_ics, but no issue or PR
7. Created flow charts
    - mapped inheritance for TM classes
        - FiniteStrain Materials (to be deprecated?) (Figure 4)
        - ComputeStress Materials (Figure 5)
        - UserObjects :: TMHardeningXX & TMPlasticXX (Figure 6)
        - traced example jacobian function (df_dintnl) to show userObject incorporation via MultiPlasticityRawComponentAssembler and _f[model]->function() (Figure 7)

8. Wiki needs to be updated
    - wiki > physics modules > tensor mechanics > plasticity and return map
        - FiniteStrainMultiPlasticity Material mentioned (should be ComputeMultiPlasticityStress?)
        - should add something about implementing custom return maps & ctos via userObjects after PR & merge

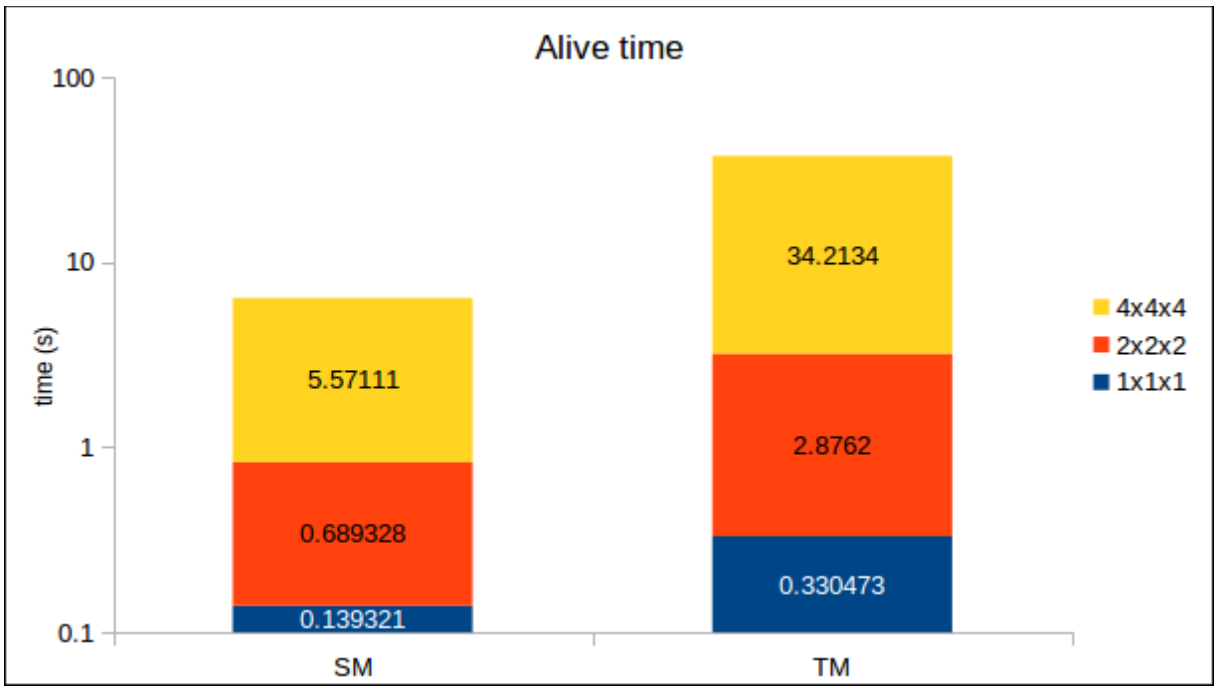


*Figure 1: Original timing, SM LSH vs TM J2*

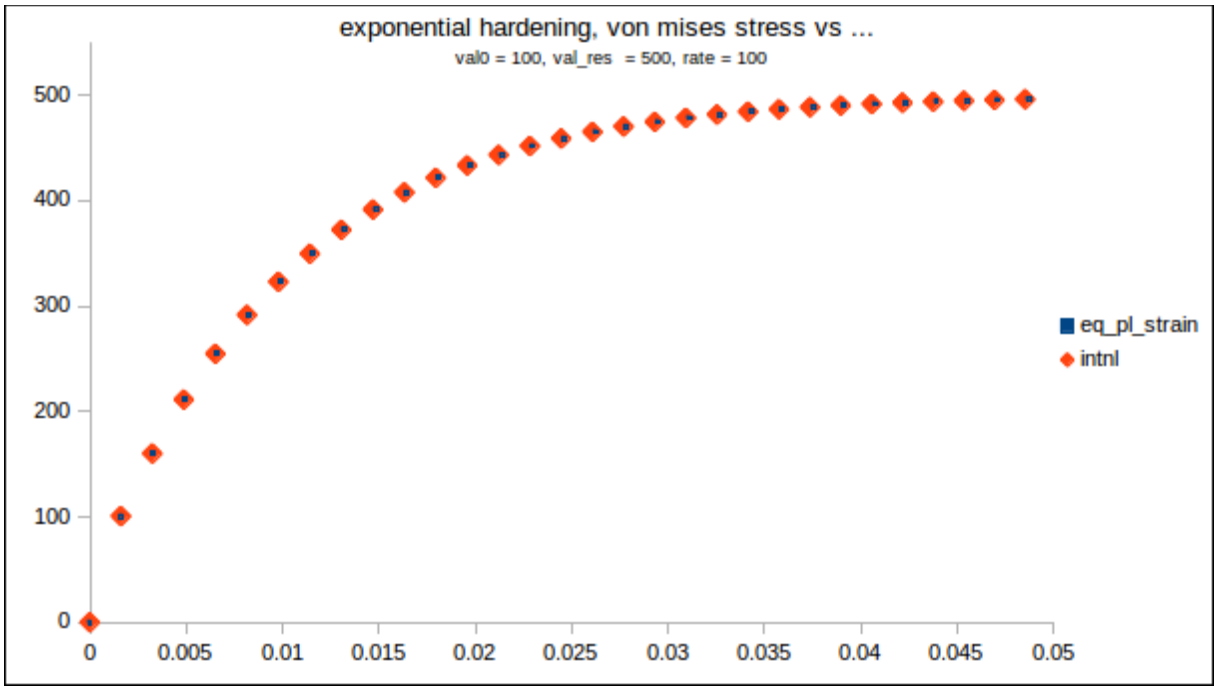


*Figure 2: Von-mises stress vs equivalent plastic strain and vs internal parameter for TM/J2 with exponential hardening*

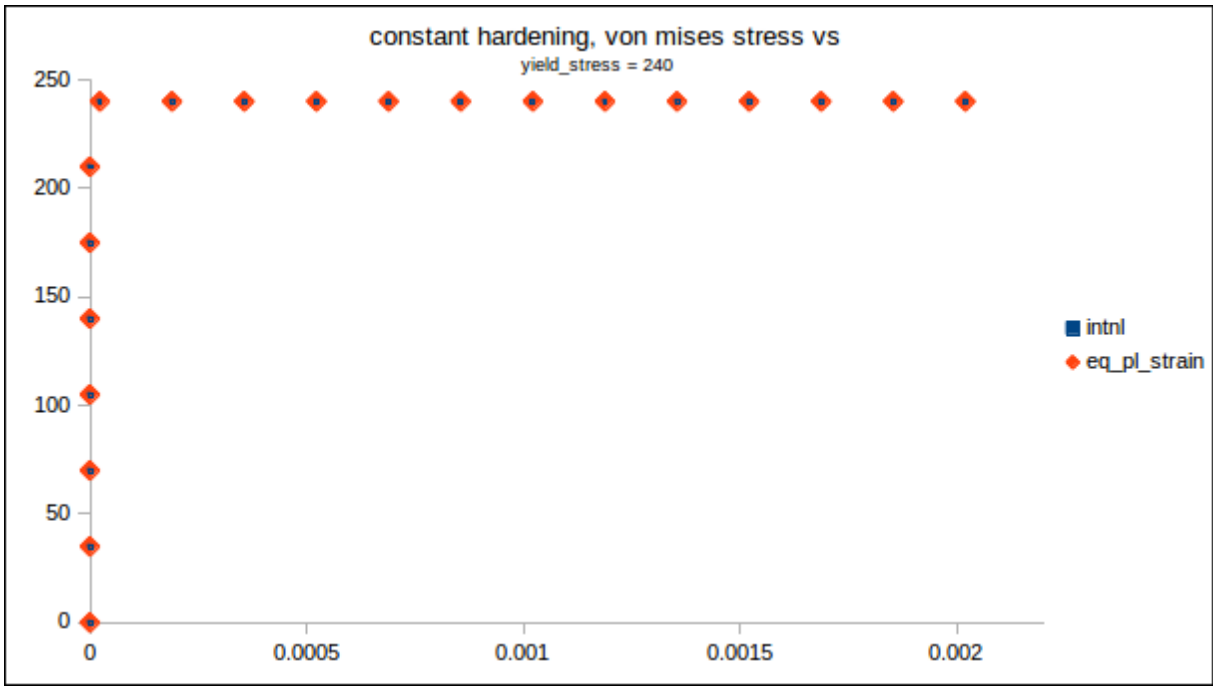


*Figure 3: Von-misses stress vs equivalent plastic strain and vs internal parameter for TM/J2 with constant hardening*

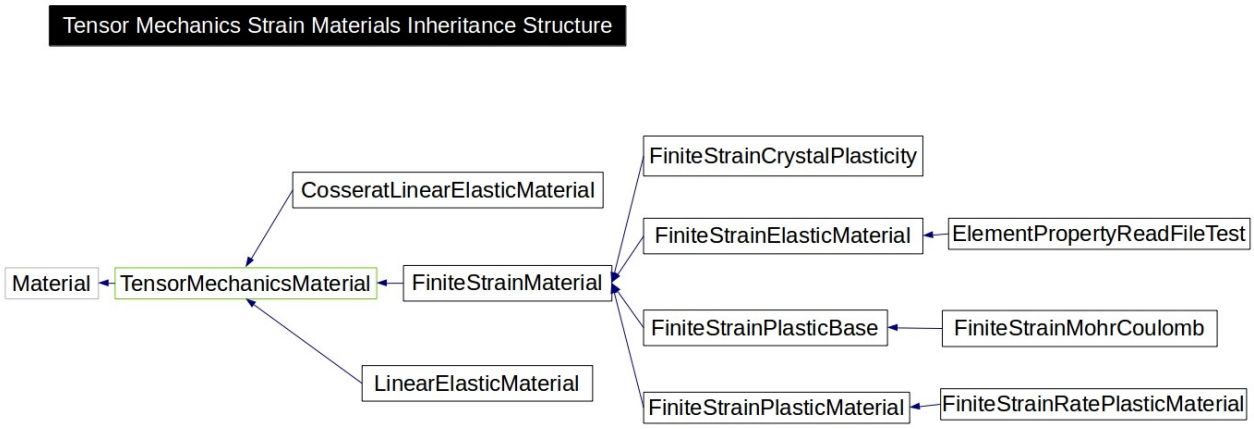


*Figure 4: Tensor mechanics FiniteStrain material class inheritance, to be deprecated in favor of userObjects and ComputeStressBase*

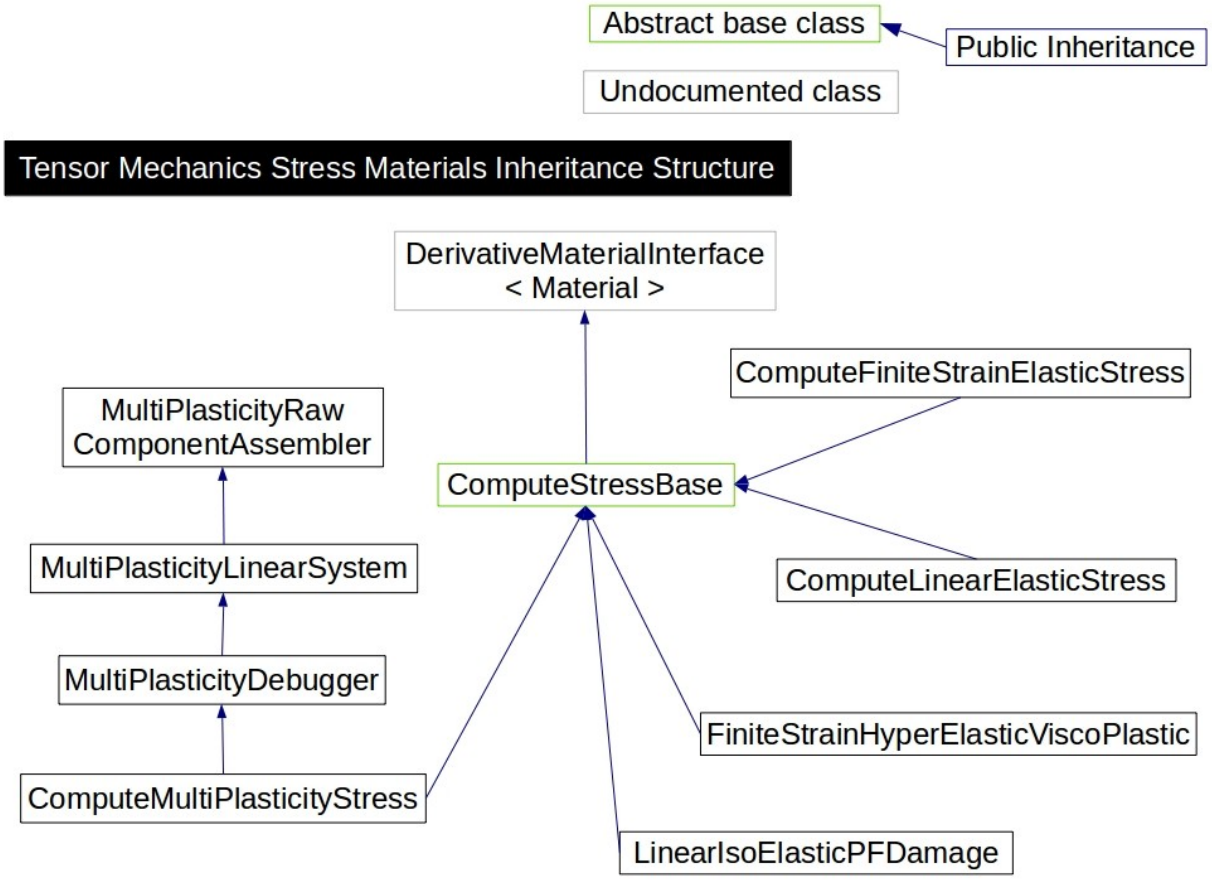


*Figure 5: Tensor mechanics ComputeStress material class inheritance, used with userObjects*

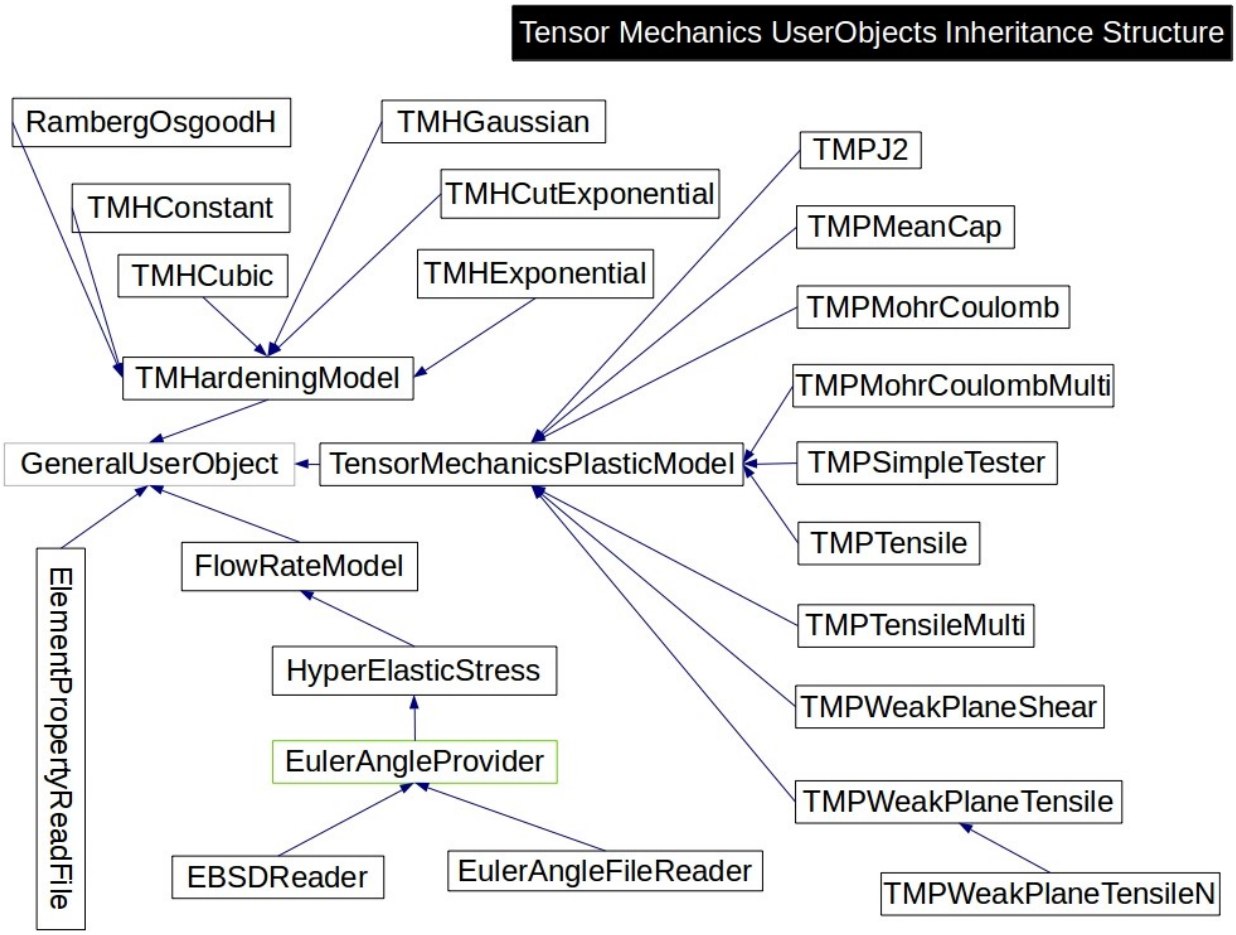


*Figure 6: Tensor mechanics userObjects class inheritance, plasticity and hardening models, used with ComputeStress materials*

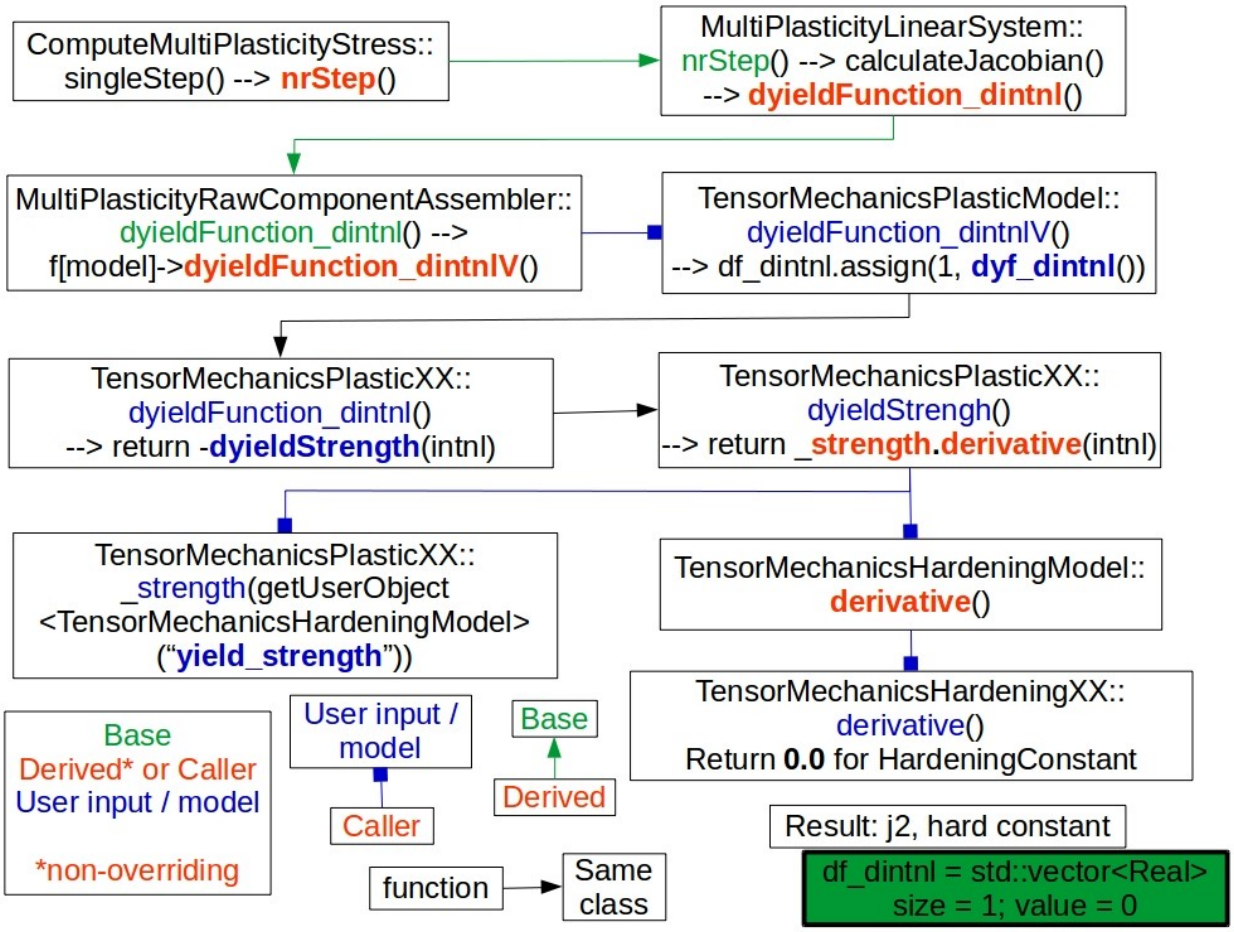


*Figure 7: Function calling to evaluate calculateJacobian() in ComputeMultiPlasticityStress::nrStep() via userObjects*

Figure 7 explanation:
1. ComputeMultiPlasticityStress::nrStep() calls calculateJacobian, which calls dyieldFunction_dintnl, which is defined in CMPS's base class, MultiPlasticityRawComponentAssembler.

2. MPRCA::dyieldFunction_dintnl calls _f[model]->dyielfFunction_dintnlV, which accesses userObjects.

3. TensorMechanicsPlsaticModel::dyieldFunction_dintnlV calls TensorMechanicsPlasticJ2::dyieldFunction_intnl.

4. J2::dyf_dintnl calls dyieldStrength, within J2.

5. J2::dyieldStrength returns _strength.derivative.

6a. _strength is the user-defined variable "yield_strength" from J2.

6b. .derivative comes from TMHardeningModel::derivative, which calls TMHardeningConstant::derivative, which returns zero.