Testing Plan - Group 5

Woodson Miles & Kelly Breedlove

- Var
  - ID()
    - make a variable, then call ID() on it and confirm the result is “0” since variable IDs start counting from 0
  - name()
    - make a variable, then call name() and confirm the result matches the name given upon construction
  - displayString()
    - make a variable, then call displayString() and confirm the result matches the name given upon construction
  - rank()
    - make a field variable, then call rank() and confirm the result is “0”
    - make a test variable, then call rank() and confirm the result is “0”
    - make a test variable with space HDIV, then call rank() and confirm the result is “1”
  - space()
    - make a field variable, then call space() and confirm the result is “L2”
    - make a test variable with space HGRAD, then call space() and confirm the result is “HGRAD”
  - varType()
    - make a test variable, then call varType() and confirm the result is “0”
    - make a field variable, then call varType() and confirm the result is “1”
    - make a trace of a field variable, use it to make a flux variable, then call varType() and confirm the result is “1”
    - make a flux variable with the other constructor, then call varType() and confirm the result is “1”
    - make a trace variable, then call varType() and confirm the result is “1”
  - op()
    - make a field variable, then call op() and confirm the result is “OP_VALUE”
    - make a test variable, then call op() and confirm the result is “OP_VALUE”
  - termTraced()
    - make a field variable, use it to make a flux variable, do a trace of it, get its IDs using varIDs(), then confirm there is only one element in the varIDs and that the one element is equal to the fieldVar’s ID
  - grad()

        - apply grad() to a variable, then call op() and confirm the result is “OP_GRAD”
    - div()
        - apply div() to a variable, then call op() and confirm the result is “OP_DIV”
    - curl()
        - apply curl(HGRAD) to a variable, then call op() and confirm the result is “OP_CURL”
    - dx()
        - apply dx() to a variable, then call op() and confirm the result is “OP_DX”
    - dy()
        - apply dy() to a variable, then call op() and confirm the result is “OP_DY”
    - x()
        - apply x() to a variable, then call op() and confirm the result is “OP_X”
    - y()
        - apply y() to a variable, then call op() and confirm the result is “OP_Y”
- VarFactory
    - testVar()
        - Make a test variable with a space HGRAD and make a second variable.
        - Use name() and space() to assert the created variable retains its given attributes.
        - Assert that the two variable are produced with different ID’s using ID().
    - fieldVar()
        - Make a field variable with a linear term and make a second variable.
        - Use name() and space() to assert the created variable retains its given attributes.
        - Assert that the two variable are produced with different ID’s using ID().
    - fluxVar()
        - Make a flux variable and make a second variable.
        - Use name() and space() to assert the created variable retains its given attributes.
        - Assert that the two variable are produced with different ID’s using ID().
    - traceVar()
        - Make a trace variable with a linear term and make a second variable.
        - Use name() and space() to assert the created variable retains its given attributes.
        - Assert that the two variable are produced with different ID’s using ID().

    - test()
        - make a test Variable and use test() and test the returned ID matches the ID of the given variable.
    - trial()
        - make a test Variable and use test() and test the returned ID matches the ID of the given variable.
    - testIDs()
        - Create three test variables with different parameters. Use testIDs() and test the returned array for the correct IDs using ID() on each test variable and comparing.
    - trialIDs()
        - Create three test variables with different parameters. Use trialIDs() and test the returned array for the correct IDs using ID() on each test variable and comparing.
    - fieldVars()
        - Create three field variables with different parameters and test that the vector returned from fieldVars()contains the same variables created with the same attributes.
    - fluxVars()
        - Create three flux variables with different parameters and test that the vector returned from fluxVars()contains the same variables created with the same attributes.
    - traceVars()
        - Create three trace variables with different parameters and test that the vector returned from traceVars()contains the same variables created with the same attributes.
- Solution
    - Solution()
        - Create two solutions using the Solution() constructor and same parameters, then confirm that they are equal and not equal to None
        - All other tests will also show that a Solution object behaves consistently
    - solve()
        - Too complex for us to test
    - addSolution()
        - Create two solutions in the same way with the Solution() constructor, then call addSolution() on one of them, compare and confirm that their L2Norms are not equal
    - clear()
        - Create a solution, addSolution() and confirm the L2NormOfSolution() is not 0, then call clear() on it and confirm that the object still exists but the L2NormOfSolution() is equal to zero since there are zero solutions
    - cubatureEnrichmentDegree() & setCubatureEnrichmentDegree()
        - Use setCubatureEnrichmentDegree() to set to x, then call cubatureEnrichmentDegree() and confirm the result is x

- L2NormOfSolution()
  - Create a solution, then call LSNormOfSolution() and confirm the result is 0 since there are 0 solutions
  - Create a solution, call addSolution(), then call LSNormOfSolution(0) and confirm the result is nearly 1.34, and call LSNormOfSolution(1) and confirm the result is nearly 216
- projectOntoMesh()
  - Create a solution, call L2NormOfSolution(0) and confirm it is equal to 0.0, then call projectOntoMesh() on it and confirm that the L2NormOfSolution(0) is no longer equal to 0.0
- energyErrorTotal()
  - not tested
- setWriteMatrixToFile()
  - not tested
- setWriteMatrixToMatrixMarketFile()
  - not tested
- setWriteRHSToMatrixMarketFile()
  - not tested
- mesh()
  - Create a solution with a specific mesh, then call mesh() and confirm that the result is the same as the mesh used to create the solution
      - The above did not work. For some reason, the object returned in mesh() is not equal the object used when constructing the solution
  - Alternately, create a solution and set the mesh to x, then call getDimension() on it and the mesh used to create it to confirm that they are equal
- setBC() & bc()
  - Use setBC() to set to x, then call bc() and confirm the result is x
    - The above did not work. For some reason, the object returned in bc() is not equal the object used in setBC()
  - Alternately, use setBC() to set to x, then call singlePointBC() on it and bc() to confirm they are equal
- setRHS() and rhs()
  - Use setRHS() to set to x, then call rhs() and confirm the result is x
    - The above did not work. For some reason, the object returned in rhs() is not equal the object used in setRHS()
  - Alternately, use setRHS() to set to x, then call nonZeroRHS() on it and rhs() to confirm they are equal
- setIP() & ip()
  - Use setIP() to set to x, then call ip() and confirm the result is x

        - The above did not work. For some reason, the object returned in ip() is not equal the object used in setIP()
    - Alternately, use setIP() to set to x, then call ip() to confirm is does not throw an excpetion
- save() & load()
    - Make a solution, call addSolution() on it, then save() and load() it to confirm it does not throw an exception
- saveToHDF5() & loadToHDF5()
    - Make a solution, call addSolution() on it, then saveToHDF5() and loadToHDF5() it to confirm it does not throw an exception
- setUseCondensedSolve()
    - Make a solution and call setUseCondensedSolve() to confirm it does not throw an exception