Cash Flow Library

Used to derive the cash flows for individual BCNs and the total cash flows for alternatives, ultimately constructing the Total Cash Flows objects

# Libraries/Classes calling on Cash Flow Library

Main

Measure

# Internal Library/Class dependencies

Discounting Library

BCN Class

Analysis Class

bcnStorage Class

# External Library Dependencies

None

# Functions

blankFlows(studyPeriod,timestepValue) – initializes a blank cash flow list to store data

bcnFlow(bcnObject,studyPeriod,timestepCount) – begins construction of cash flows for a given BCN

bcnFlowNonRecur(bcnObject,discountRate) – completes construction of flows for non-recurring BCNs

bcnFlowRecur(bcnObject,discountRate,timestep) – completes construction of flows for recurring BCNs

rvCalc(bcnObject,value,*timestep*) – calculates the residual value of a BCN

totalFlows(bcnObjectList,altID) – calculates the total flows for an alternative

# Pseudo Code

Optional inputs are in italics

Begin Pseudocode

Import BCN Class

Import Analysis Class

Import discounting library

Import BCN Storage Class

**blankFlow(studyPeriod,timestepValue)**

timestepCount = studyPeriod/timestepValue

return numpy.zeros(timestepCount + 1)

**end blankFlow**

**bcnFlow(discountRate,bcnObject,studyPeriod,timestepCount)**

if recurBool == false

bcnFlowNonRecur(bcnObject,discountRate)

else

bcnFlowRecur(bcnObject,discountRate)

**bcnFlowNonRecur(discountRate,bcnObject,discountRate)**

bcnFlowNonDisc = blankFlow(studyPeriod,timestepValue)

bcnFlowDisc = blankFlow(studyPeriod,timestepValue)

quantList = blankFlow(studyPeriod,timestepValue)

if valuePerQ is blank

return bcnFlowNonDisc, bcnFlowDisc

if *quantVarValue* is not blank

*quantEsc* = escalatedQuantCalc(quantity,quantVarRateType,quantVarValue,initialOcc)

add quantEsc to the element in quantList corresponding to the appropriate timestep

*value* = quantityEsc\*valuePerQ

Else

add quantity to the element in quantList corresponding to the appropriate timestep

*value* = quantity\*valuePerQ

*discMult* = discounting.spv (initialOcc, recurVarValue,discountRate)

*discValue* = discounting.discValueCalc(value, *discMult*)

For the index corresponding to the timestep the payment occurs in based on the initialOcc, add *value* to bcnFlowNonDisc and add *discValue* to bcnFlowDisc

If rvBool == True

Call to rvCalc(bcnObject,value)

residValue = rvCalc(bcnObject,value)

residValueMult = discounting.spv (studyPeriod, recurVarValue,discountRate)

residValueDisc = discounting.discValueCalc(residValue, residValueMult)

For the index corresponding to the final timestep add *residValue* to bcnFlowNonDisc and add *residValueDisc* to bcnFlowDisc

Return bcnFlowNonDisc, bcnFlowDisc and quantList

**End bcnFlowNonRecur**

**bcnFlowRecur(discountRate,bcnObject,discountRate, timestep)**

bcnFlowNonDisc = blankFlow(studyPeriod,timestepValue)

bcnFlowDisc = blankFlow(studyPeriod,timestepValue)

recurList = blankFlow(studyPeriod,timestepValue)

quantList = blankFlow(studyPeriod,timestepValue)

Using the initialOcc, recurInterval, endDate (if one exists), timestep, and studyPeriod, place a one in each list element where the bcn will have calculate value (*recurList*), for instance if:

initialOcc = 2,

recurInterval = 2

endDate = 6

timestep = 1

studyPeriod = 10

then the *recurList* would look like the following

time [0,1,2,3,4,5,6,7,8,9,10]

[0,0,1,0,1,0,1,0,0,0,0]

if valuePerQ is blank

return bcnFlowNonDisc, bcnFlowDisc

if *quantVarValue* is not blank:

for each timestep corresponding to a “1” in recurList

*quantEsc* = escalatedQuantCalc(quantity,quantVarRateType,quantVarValue,initialOcc,timestep)

*value* = quantEsc\*valuePerQ

For the index corresponding to the timestep the quantity occurs add quantEsc to quantList and value to bcnFlowNonDisc

Else

quantList = quantList + quantity\*recurList

bcnFlowNonDisc = bcnFlowNonDisc + quantity\*valuePerQ\*recurLst

For each timestep corresponding to a “1” in recurList

discMult = discounting.spv(timestep, recurrenceVarVal,discountRate)

discValue = discounting.discValueCalc(value, *discMult,bcn.recurRateVal*)

For the index corresponding to the timestep the payment occurs, add *discValue* to bcnFlowDisc

If rvBool == True

Call to rvCalc(bcnObject,value)

residValue = rvCalc(bcnObject,value)

residValueMult = discounting.spv (studyPeriod, recurVarValue,discountRate)

residValueDisc = discounting.discValueCalc(residValue, residValueMult)

For the index corresponding to the final timestep add *residValue* to bcnFlowNonDisc and add *residValueDisc* to bcnFlowDisc

Return bcnFlowNonDisc, bcnFlowDisc and quantList

**End bcnFlowRecur**

**rvCalc(bcnObject,value,*timestep*)**

Find remaining service life at end of study period

If recurBool == False

remainingLife = bcnLife – (studyPeriod – initialOcc)

Else

remaininigLife = bcnLife – (studyPeriod – most recent occurrence date)

if remainingLife <= 0

return 0

else

return residualValue = remainingLife/bcnLife\*value

**End rvCalc**

**totalFlows(bcnStorageObjectList,altID)**

(Items numbered as Item1 through Item20)

{totCostNonDisc} = blankFlow(studyPeriod,timestepValue)

{totCostDisc} = blankFlow(studyPeriod,timestepValue)

{totCostNonDiscInv} = blankFlow(studyPeriod,timestepValue)

{totCostDiscInv} = blankFlow(studyPeriod,timestepValue)

{totCostNonDiscNonInv} = blankFlow(studyPeriod,timestepValue)

{totCostDiscNonInv} = blankFlow(studyPeriod,timestepValue)

{totBenefitsNonDisc} = blankFlow(studyPeriod,timestepValue)

{totBenefitsDisc} = blankFlow(studyPeriod,timestepValue)

{totCostDir} = blankFlow(studyPeriod,timestepValue)

{totCostInd} = blankFlow(studyPeriod,timestepValue)

{totCostExt} = blankFlow(studyPeriod,timestepValue)

{totCostDirDisc} = blankFlow(studyPeriod,timestepValue)

{totCostIndDisc} = blankFlow(studyPeriod,timestepValue)

{totCostExtDisc} = blankFlow(studyPeriod,timestepValue)

{totBenefitsDir} = blankFlow(studyPeriod,timestepValue)

{totBenefitsInd} = blankFlow(studyPeriod,timestepValue)

{totBenefitsExt} = blankFlow(studyPeriod,timestepValue)

{totBenefitsDirDisc} = blankFlow(studyPeriod,timestepValue)

{totBenefitsIndDisc} = blankFlow(studyPeriod,timestepValue)

{totBenefitsExtDisc} = blankFlow(studyPeriod,timestepValue)

Loop through bcnStorageObjectList

if altID in bcnStorage.altIID

For bcnStorageObject in bcnStorageObjectList:

If bcnStorage.sensBool == false and bcnStorage.uncBool == false

bcnFlowNonDisc = bcnStorage.bcnNonDiscFlow

bcnFlowDisc = bcnStorage.bcnDiscFlow

quantList = bcnStorage.quantList

else if bcnStorage.sensBool == true

bcnFlowNonDisc = bcnStorage.sensFlowNonDisc

bcnFlowDisc = bcnStorage.sensFlowDisc

quantList = bcnStorage.sensQuantList

else if bcnStorage.uncBool == true

bcnFlowNonDisc = bcnStorage.UncFlowNonDisc

bcnFlowDisc = bcnStorage.uncFlowDisc

quantList = bcnStorage.uncQuantList

If bcnStorage.type == cost

Add bcnFlowNonDisc to totCostNonDisc element by element

Add bcnFlowDisc to totCostDisc element by element

If bcnInvestBool == True

Add bcnFlowNonDisc to totCostNonDiscInv element by element

Add bcnFlowDisc to totCostDiscInv element by element

Else

Add bcnFlowNonDisc to totCostNonDiscNonInv element by element

Add bcnFlowDisc to totCostDiscNonInv element by element

If bcnStorage.subtype == Dir

Add bcnFlowNonDisc to totCostDir element by element

Add bcnFlowDisc to totCostDirDisc element by element

Else if bcnStorage.subtype == Ind

Add bcnFlowNonDisc to totCostInd element by element

Add bcnFlowDisc to totCostIndDisc element by element

Else if bcnStorage.subtype == Ext

Add bcnFlowNonDisc to totCostExt element by element

Add bcnFlowDisc to totCostExtDisc element by element

Else if bcnStorage.type == “Benefit”

Add bcnFlowNonDisc to totBenefitsNonDisc element by element

Add bcnFlowDisc to totBenefitsDisc element by element

If bcnInvestBool == True

Add bcnFlowNonDisc to totBenefitsNonDiscInv element by element

Add bcnFlowDisc to totBenefitsDiscInv element by element

Else

Add bcnFlowNonDisc to totBenefitsNonDiscNonInv element by element

Add bcnFlowDisc to totBenefitsDiscNonInv element by element

If bcnStorage.subtype == Dir

Add bcnFlowNonDisc to totBenefitsDir element by element

Add bcnFlowDisc to totBenefitsDirDisc element by element

Else if bcnStorage.subtype == Ind

Add bcnFlowNonDisc to totBenefitsInd element by element

Add bcnFlowDisc to totBenefitsIndDisc element by element

Else if bcnStorage.subtype == Ext

Add bcnFlowNonDisc to totBenefitsExt element by element

Add bcnFlowDisc to totBenefitsExtDisc element by element

If bcn.tag exists or type is “non-monetary”

if bcnStorage.sensBool == false and bcnStorage.uncBool == false

If no totalOptionalFlows object exists for bcn.tag

Initialize totalOptionalFlows object via

totalOptionalFlows(altID,sensBool,bcnType,bcnSubType,bcnTag,bcnFlowDisc, quantList)

else if bcnStorage.sensBool == true

if totalOptionalFlows object with sensBool == True does Not exist

If no totalOptionalFlows object exists for bcn.tag

Initialize totalOptionalFlows object via

totalOptionalFlows(altID,sensBool,bcnType,bcnSubType,bcnTag,bcnFlowDisc, quantList)

else if bcnStorage.uncBool == true

not implemented yet

else if totalOptionalFlows object already exists for tag/sensBool or tag/uncBool combination

update totalOptionalFlows object for the given tag/sensBool combination by adding bcnFlowDisc and quantList to the values currently in the totalOptionalFlows object

if bcnStorage.sensBool == false and bcnStorage.uncBool == false

Construct Total Required Flows object

totalRequiredFlows(altID,Item1,…,Item20,baseBool)

else if bcnStorage.sensBool == true

if totalRequiredFlows objects with sensBool == True do NOT exist

Construct Total Required Flows object

totalRequiredFlows(altID,sensBool,Item1,…,Item20,baseBool)

if totalRequiredFlows objects with sensBool == True do exist

Update Total Required Flows objects

Loop through totalRequiredFlows objects and

totalRequiredFlows.update(Item1,…,Item20)

else if bcnStorage.sensBool == true

not implemented yet

return nothing

**end totalFlows**