OSCEF APIs

Generated by Doxygen 1.7.6.1

Thu Aug 29 2013 10:51:34
4.1.3.10 getAmountMax . . . . . . . . . . . . . . . . . . . . . . 15
4.1.3.11 getClassLabel . . . . . . . . . . . . . . . . . . . . . . 15
4.1.3.12 getComment . . . . . . . . . . . . . . . . . . . . . . . 15
4.1.3.13 getExpense . . . . . . . . . . . . . . . . . . . . . . . 15
4.1.3.14 getGamma . . . . . . . . . . . . . . . . . . . . . . . . 15
4.1.3.15 getLabel . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.1.3.16 getNodeNo . . . . . . . . . . . . . . . . . . . . . . . . 16
4.1.3.17 getParent . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.1.3.18 getParentPtr . . . . . . . . . . . . . . . . . . . . . . . 16
4.1.3.19 getPrice . . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.1.3.20 getRho . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.1.3.21 getSigma . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.1.3.22 getSize . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.1.3.23 getTax . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.1.3.24 getVars . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.1.3.25 isAbstractOutputs . . . . . . . . . . . . . . . . . . . . 18
4.1.3.26 isCustomersFactorsIncome . . . . . . . . . . . . . . . 18
4.1.3.27 isCustomersIncome . . . . . . . . . . . . . . . . . . . 18
4.1.3.28 isCustomersInvestmentIncome . . . . . . . . . . . . . 18
4.1.3.29 isCustomersPDTS . . . . . . . . . . . . . . . . . . . . . 19
4.1.3.30 isCustomersProducts . . . . . . . . . . . . . . . . . . . 19
4.1.3.31 isCustomersProfits . . . . . . . . . . . . . . . . . . . . 19
4.1.3.32 isCustomersSavings . . . . . . . . . . . . . . . . . . . . 19
4.1.3.33 isCustomersUtility . . . . . . . . . . . . . . . . . . . . 19
4.1.3.34 isDomestic . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.1.3.35 isInput . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.1.3.36 isInputDomestic . . . . . . . . . . . . . . . . . . . . . . 20
4.1.3.37 isInputFactor . . . . . . . . . . . . . . . . . . . . . . . 20
4.1.3.38 isInputImported . . . . . . . . . . . . . . . . . . . . . . 20
4.1.3.39 isOutput . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.1.3.40 isOutputFinal . . . . . . . . . . . . . . . . . . . . . . . . 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Class Reference</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8.3.2</td>
<td>getOutCode</td>
<td>54</td>
</tr>
<tr>
<td>4.8.3.3</td>
<td>getWeight</td>
<td>54</td>
</tr>
<tr>
<td>4.8.3.4</td>
<td>isValid</td>
<td>55</td>
</tr>
<tr>
<td>4.8.3.5</td>
<td>setInCode</td>
<td>55</td>
</tr>
<tr>
<td>4.8.3.6</td>
<td>setOutCode</td>
<td>55</td>
</tr>
<tr>
<td>4.8.3.7</td>
<td>setWeight</td>
<td>55</td>
</tr>
<tr>
<td>4.9</td>
<td>Margins Class Reference</td>
<td>56</td>
</tr>
<tr>
<td>4.9.1</td>
<td>Detailed Description</td>
<td>56</td>
</tr>
<tr>
<td>4.9.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>56</td>
</tr>
<tr>
<td>4.9.2.1</td>
<td>Margins</td>
<td>56</td>
</tr>
<tr>
<td>4.10</td>
<td>OscefException Class Reference</td>
<td>57</td>
</tr>
<tr>
<td>4.10.1</td>
<td>Detailed Description</td>
<td>58</td>
</tr>
<tr>
<td>4.10.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>58</td>
</tr>
<tr>
<td>4.10.2.1</td>
<td>OscefException</td>
<td>58</td>
</tr>
<tr>
<td>4.10.3</td>
<td>Member Function Documentation</td>
<td>58</td>
</tr>
<tr>
<td>4.10.3.1</td>
<td>display</td>
<td>58</td>
</tr>
<tr>
<td>4.10.3.2</td>
<td>getFilename</td>
<td>59</td>
</tr>
<tr>
<td>4.10.3.3</td>
<td>getLineNo</td>
<td>59</td>
</tr>
<tr>
<td>4.10.3.4</td>
<td>getMessage</td>
<td>59</td>
</tr>
<tr>
<td>4.10.3.5</td>
<td>getOscefError</td>
<td>59</td>
</tr>
<tr>
<td>4.10.3.6</td>
<td>setFilename</td>
<td>59</td>
</tr>
<tr>
<td>4.10.3.7</td>
<td>setLineNum</td>
<td>60</td>
</tr>
<tr>
<td>4.10.3.8</td>
<td>setMessage</td>
<td>60</td>
</tr>
<tr>
<td>4.10.3.9</td>
<td>what</td>
<td>60</td>
</tr>
<tr>
<td>4.11</td>
<td>PdnFcn Class Reference</td>
<td>60</td>
</tr>
<tr>
<td>4.11.1</td>
<td>Detailed Description</td>
<td>63</td>
</tr>
<tr>
<td>4.11.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>63</td>
</tr>
<tr>
<td>4.11.2.1</td>
<td>PdnFcn</td>
<td>63</td>
</tr>
<tr>
<td>4.11.2.2</td>
<td>PdnFcn</td>
<td>64</td>
</tr>
<tr>
<td>4.11.2.3</td>
<td>PdnFcn</td>
<td>64</td>
</tr>
<tr>
<td>4.11.2.4</td>
<td>PdnFcn</td>
<td>64</td>
</tr>
</tbody>
</table>
4.13.3.18 extractSectors ............................................. 90
4.13.3.19 factorAggregate ........................................... 90
4.13.3.20 factorAggregate ........................................... 91
4.13.3.21 factorPermuted ........................................... 91
4.13.3.22 flip .......................................................... 91
4.13.3.23 generateHeaders ........................................... 92
4.13.3.24 getAutoSparsify .......................................... 92
4.13.3.25 getCode ............................................. 92
4.13.3.26 getColSum .................................................. 93
4.13.3.27 getFactors ............................................... 93
4.13.3.28 getHeader .................................................. 93
4.13.3.29 getMargins ............................................. 94
4.13.3.30 getMatrix .............................................. 94
4.13.3.31 getMatSum .............................................. 94
4.13.3.32 getNnz ................................................... 95
4.13.3.33 getNnzPercent ......................................... 95
4.13.3.34 getQ_RM ................................................. 95
4.13.3.35 getRasMaxIter .......................................... 96
4.13.3.36 getRasTol .............................................. 96
4.13.3.37 getRegions ............................................. 96
4.13.3.38 getRowSum ............................................. 96
4.13.3.39 getSamDim ............................................. 97
4.13.3.40 getSectors ............................................. 97
4.13.3.41 getSize .................................................. 97
4.13.3.42 getSparseMatrix ....................................... 97
4.13.3.43 getSparseSubmatrix .................................. 98
4.13.3.44 getSparseSubmatrix .................................. 98
4.13.3.45 getSubmatrix .......................................... 98
4.13.3.46 getValue ................................................ 99
4.13.3.47 getYear ................................................ 99
4.13.3.48 isBalanced ............................................. 100
<p>| 4.13.3.49 | isCodeValid | 100 |
| 4.13.3.50 | isColCoeffMat | 101 |
| 4.13.3.51 | isColSumsValid | 101 |
| 4.13.3.52 | isDense | 101 |
| 4.13.3.53 | isHeadersValid | 102 |
| 4.13.3.54 | isMatSumsValid | 102 |
| 4.13.3.55 | isMatValid | 102 |
| 4.13.3.56 | isNnzBlocksValid | 102 |
| 4.13.3.57 | isNonNegative | 103 |
| 4.13.3.58 | isRowSumsValid | 103 |
| 4.13.3.59 | isSizeValid | 103 |
| 4.13.3.60 | isSparse | 103 |
| 4.13.3.61 | isTaxBlock | 104 |
| 4.13.3.62 | isValid | 104 |
| 4.13.3.63 | isYearValid | 104 |
| 4.13.3.64 | marginAggregate | 104 |
| 4.13.3.65 | marginAggregate | 105 |
| 4.13.3.66 | marginPermute | 105 |
| 4.13.3.67 | multiply | 106 |
| 4.13.3.68 | operator!= | 106 |
| 4.13.3.69 | operator* | 107 |
| 4.13.3.70 | operator+ | 107 |
| 4.13.3.71 | operator== | 108 |
| 4.13.3.72 | permuteBlock10toMR | 108 |
| 4.13.3.73 | permuteBlock10toRM | 109 |
| 4.13.3.74 | print | 109 |
| 4.13.3.75 | regionAggregate | 109 |
| 4.13.3.76 | regionAggregate | 109 |
| 4.13.3.77 | regionDisaggregate | 110 |
| 4.13.3.78 | regionDisaggregate | 111 |
| 4.13.3.79 | regionPermute | 111 |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>resetRasMaxIter</td>
<td>112</td>
</tr>
<tr>
<td>resetRasTol</td>
<td>112</td>
</tr>
<tr>
<td>resize</td>
<td>112</td>
</tr>
<tr>
<td>sectorAggregate</td>
<td>113</td>
</tr>
<tr>
<td>sectorAggregate</td>
<td>113</td>
</tr>
<tr>
<td>sectorDisaggregate</td>
<td>114</td>
</tr>
<tr>
<td>sectorDisaggregate</td>
<td>114</td>
</tr>
<tr>
<td>sectorPermute</td>
<td>115</td>
</tr>
<tr>
<td>setAutoSparsify</td>
<td>115</td>
</tr>
<tr>
<td>setCode</td>
<td>115</td>
</tr>
<tr>
<td>setColSum</td>
<td>116</td>
</tr>
<tr>
<td>setFactors</td>
<td>116</td>
</tr>
<tr>
<td>setHeader</td>
<td>116</td>
</tr>
<tr>
<td>setMargins</td>
<td>117</td>
</tr>
<tr>
<td>setMatrix</td>
<td>117</td>
</tr>
<tr>
<td>setQ_RM</td>
<td>117</td>
</tr>
<tr>
<td>setRasMaxIter</td>
<td>117</td>
</tr>
<tr>
<td>setRasTol</td>
<td>118</td>
</tr>
<tr>
<td>setRegions</td>
<td>118</td>
</tr>
<tr>
<td>setRowSum</td>
<td>118</td>
</tr>
<tr>
<td>setSamDim</td>
<td>119</td>
</tr>
<tr>
<td>setSectors</td>
<td>119</td>
</tr>
<tr>
<td>setSparseMatrix</td>
<td>119</td>
</tr>
<tr>
<td>setSparseSubmatrix</td>
<td>119</td>
</tr>
<tr>
<td>setSparseSubmatrix</td>
<td>120</td>
</tr>
<tr>
<td>setSparseSubmatrix</td>
<td>120</td>
</tr>
<tr>
<td>setValue</td>
<td>120</td>
</tr>
<tr>
<td>setYear</td>
<td>121</td>
</tr>
<tr>
<td>simulateRandomSam</td>
<td>121</td>
</tr>
<tr>
<td>toVer8</td>
<td>122</td>
</tr>
<tr>
<td>transformFactors</td>
<td>122</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.13.3.11 transformMargins</td>
<td>122</td>
</tr>
<tr>
<td>4.13.3.11 transformRegions</td>
<td>123</td>
</tr>
<tr>
<td>4.13.3.11 transformSectors</td>
<td>123</td>
</tr>
<tr>
<td>4.13.3.11 write</td>
<td>123</td>
</tr>
<tr>
<td>4.13.3.11 writeLists</td>
<td>124</td>
</tr>
<tr>
<td>4.14 SamDim Class Reference</td>
<td>124</td>
</tr>
<tr>
<td>4.14.1 Detailed Description</td>
<td>127</td>
</tr>
<tr>
<td>4.14.2 Constructor &amp; Destructor Documentation</td>
<td>127</td>
</tr>
<tr>
<td>4.14.2.1 SamDim</td>
<td>127</td>
</tr>
<tr>
<td>4.14.2.2 SamDim</td>
<td>128</td>
</tr>
<tr>
<td>4.14.2.3 SamDim</td>
<td>128</td>
</tr>
<tr>
<td>4.14.3 Member Function Documentation</td>
<td>128</td>
</tr>
<tr>
<td>4.14.3.1 computeIndices</td>
<td>128</td>
</tr>
<tr>
<td>4.14.3.2 getBlockHeader</td>
<td>128</td>
</tr>
<tr>
<td>4.14.3.3 getBlockIndex</td>
<td>129</td>
</tr>
<tr>
<td>4.14.3.4 getHeader</td>
<td>129</td>
</tr>
<tr>
<td>4.14.3.5 getI1</td>
<td>129</td>
</tr>
<tr>
<td>4.14.3.6 getI2</td>
<td>130</td>
</tr>
<tr>
<td>4.14.3.7 getJ1</td>
<td>130</td>
</tr>
<tr>
<td>4.14.3.8 getJ2</td>
<td>130</td>
</tr>
<tr>
<td>4.14.3.9 getL</td>
<td>131</td>
</tr>
<tr>
<td>4.14.3.10 isFactor</td>
<td>131</td>
</tr>
<tr>
<td>4.14.3.11 isMargin</td>
<td>131</td>
</tr>
<tr>
<td>4.14.3.12 isRegion</td>
<td>131</td>
</tr>
<tr>
<td>4.14.3.13 isSector</td>
<td>132</td>
</tr>
<tr>
<td>4.14.3.14 operator!=</td>
<td>132</td>
</tr>
<tr>
<td>4.14.3.15 operator==</td>
<td>132</td>
</tr>
<tr>
<td>4.14.3.16 setHeader</td>
<td>133</td>
</tr>
<tr>
<td>4.14.3.17 setL</td>
<td>133</td>
</tr>
<tr>
<td>4.14.3.18 setSFRMC</td>
<td>133</td>
</tr>
<tr>
<td>4.14.3.19 setVersion</td>
<td>134</td>
</tr>
</tbody>
</table>
4.14.3.20 setYear ........................................ 134
4.14.4 Member Data Documentation .......................... 134
  4.14.4.1 expectedPattern .............................. 134
  4.14.4.2 isSumPositive ............................... 135
4.15 Sams Class Reference ................................. 135
  4.15.1 Detailed Description ................................ 137
  4.15.2 Constructor & Destructor Documentation ............ 137
    4.15.2.1 Sams ...................................... 137
  4.15.3 Member Function Documentation .................... 138
    4.15.3.1 areBalanced .................................. 138
    4.15.3.2 balanceByRas ................................ 138
    4.15.3.3 factorAggregate ................................ 139
    4.15.3.4 factorAggregate ................................ 139
    4.15.3.5 getOutOfCore .................................. 139
    4.15.3.6 getSam ....................................... 139
    4.15.3.7 getSam ....................................... 140
    4.15.3.8 getSamDim .................................... 140
    4.15.3.9 getSize ....................................... 140
    4.15.3.10 isValid ...................................... 140
    4.15.3.11 marginAggregate .............................. 141
    4.15.3.12 marginAggregate .............................. 141
    4.15.3.13 operator!= ................................... 142
    4.15.3.14 operator== ................................... 142
    4.15.3.15 read ......................................... 142
    4.15.3.16 regionAggregate .............................. 143
    4.15.3.17 regionAggregate .............................. 143
    4.15.3.18 regionDisaggregate ........................... 143
    4.15.3.19 regionDisaggregate ........................... 144
    4.15.3.20 sectorAggregate .............................. 144
    4.15.3.21 sectorAggregate .............................. 145
    4.15.3.22 sectorDisaggregate ........................... 145
4.15.3.23 sectorDisaggregate .............................. 145
4.15.3.24 setSam .......................................... 146
4.15.3.25 setSamDim ...................................... 146
4.15.3.26 write ............................................. 146

4.16 Sectors Class Reference ................................. 146
  4.16.1 Detailed Description .............................. 147
  4.16.2 Constructor & Destructor Documentation .......... 147
    4.16.2.1 Sectors ...................................... 147

4.17 Utilities Class Reference ............................... 148
  4.17.1 Detailed Description .............................. 148

5 Example Documentation ................................... 149
  5.1 Check .................................................. 149
Chapter 1

OSCEF APIs

The CIM-EARTH (Community Integrated Model for Energy and Resource Trajectories for Humankind) framework is an environment for economic modeling and simulation. The first version of CIM-EARTH is written in AMPL. To facilitate the adoption of the CIM-EARTH framework and to further develop data processing and analysis services, we are moving the implementation from AMPL to C++ and adopting standard input and output formats to produce the Open-Source CIM-EARTH Framework (OSCEF). The core of OSCEF is the methodology for formulating and solving large-scale computable general equilibrium (CGE) models.

Version

1.00

Authors

Sou-Cheng Choi, sctchoi@ci.uchicago.edu
Todd Munson, tmunson@mcs.anl.gov

Creation date: Sep 26, 2011
Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CesNode ......................................................... 7
CustUtil ......................................................... 32
PdnFcn ......................................................... 60
Cge ............................................................... 27
List ............................................................. 43
Cgds .............................................................. 26
Factors ....................................................... 42
Margins ....................................................... 56
Regions ....................................................... 74
Sectors ....................................................... 146
Mapper ......................................................... 47
MapperEntry ................................................ 53
OscefException ........................................... 57
Sam ............................................................ 75
SamDim ....................................................... 124
Sams .......................................................... 135
Utilities ....................................................... 148
Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- **CesNode**
  A constant-elasticity-of-substitution node ........................................... 7

- **Cgds**
  A number-indexed list of cgds ................................................................. 26

- **Cge**
  Class for constructing a CGE (computational general equilibrium) problem ................................................................. 27

- **CustUtil**
  Class for constructing a customer-utility tree ......................................... 32

- **Factors**
  A number-indexed list of factors ............................................................... 42

- **List**
  Parent class for a numbered list of codes ............................................... 43

- **Mapper**
  A collection of tuples from the class MapperEntry ................................... 47

- **MapperEntry**
  A tuple of in-code, out-code, and weight ............................................... 53

- **Margins**
  A number-indexed list of margins ........................................................... 56

- **OscefException**
  Throw exceptions for all core classes in the OSCEF framework ................ 57

- **PdnFcn**
  Class for reading an input file of standardized production function .......... 60

- **Regions**
  A number-indexed list of regions ............................................................. 74
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Social accounting matrix in standardized dense or sparse formats</td>
<td>75</td>
</tr>
<tr>
<td>SamDim</td>
<td>This class contains the structure information of submatrices in a SAM</td>
<td>124</td>
</tr>
<tr>
<td>Sams</td>
<td>Social accounting matrices in standardized dense or sparse formats</td>
<td>135</td>
</tr>
<tr>
<td>Sectors</td>
<td>A number-indexed list of sectors</td>
<td>146</td>
</tr>
<tr>
<td>Utilities</td>
<td>Utilities or tools as inline functions</td>
<td>148</td>
</tr>
</tbody>
</table>
Chapter 4

Class Documentation

4.1 CesNode Class Reference

A constant-elasticity-of-substitution node.
#include <CesNode.h>

Inheritance diagram for CesNode:

```
CesNode
  CustUtil  PdnFcn
```

Public Member Functions

- CesNode (const int=0, const string=string(), const string=string(), const string=string())
  CesNode Constructor.
- CesNode (const int, const string, const string, const string, const double, const double=0, const double=1, const double=0)
  CesNode Constructor.
- virtual ~CesNode ()
  CesNode default destructor.
- virtual string getClassLabel () const
  Gets the class label of this entry.
- virtual string getLabel () const
Gets the label of this entry.
• virtual string getComment () const
  Gets a comment of this entry.
• virtual double getAlpha () const
  Gets the share parameter of this (input) entry.
• virtual double getGamma () const
  Gets the efficiency parameter of this (input) entry.
• virtual double getSigma () const
  Gets the elasticity of substitution of this entry.
• virtual double getRho () const
  Gets rho = 1 - 1/sigma where sigma is the elasticity of substitution of this entry.
• virtual double getAmount () const
  Gets the amount of this (input/output) entry.
• virtual double getAmountMax () const
  Gets the maximum amount of this (input/output) entry.
• virtual double getPrice () const
  Gets the unit price of this entry.
• virtual double getExpense () const
  Gets the expenditure associated with this entry.
• virtual double getTax () const
  Gets the tax of this entry.
• virtual void setClassLabel (const string)
  Sets the class label of this entry.
• virtual void setLabel (const string)
  Sets label of this entry.
• virtual void setComment (const string)
  Sets a comment of this entry.
• virtual void setAlpha (const double)
  Sets the share parameter of this (input) entry.
• virtual void setGamma (const double)
  Sets the efficiency parameter of this (input) entry.
• virtual void setSigma (const double, const bool=true)
  Sets the elasticity of substitution of this (input) entry. It also updates rho.
• virtual void setAmount (const double)
  Sets the amount of this (input/output) entry.
• virtual void setAmountMax (const double)
  Sets the maximum amount of this (input/output) entry.
• virtual void setPrice (const double)
  Sets the price of this (input/output) entry.
• virtual void setExpense (const double)
4.1 CesNode Class Reference

Sets the expenditure of this (input/output) entry.
• virtual void setTax (const double)

Sets the tax of this (input/final output) entry.
• virtual bool operator== (CesNode &)

Overload operator == for comparing two CesNodes (non-recursively).
• virtual bool operator!= (CesNode &)

Overload operator != for comparing two CesNode instances.
• virtual void add (const CesNode &)

Add a child node.
• virtual CesNode & get (const size_t)

Gets the child node by its index value.
• virtual CesNode & get (const string, const bool=true)

Gets a descendant by its code.
• virtual CesNode & get (bool &, const string, const bool=true)

Gets a descendant by its code.
• virtual size_t getSize (const bool=true)

Returns the number of children or grandchildren.
• virtual void printNode ()
• virtual void print (std::ostream &)
• virtual std::string printstr (CesNode &)
• virtual bool isOutput () const

Determines if a node represents an output.
• virtual bool isOutputImmediate () const

Determines if a node represents an immediate output.
• virtual bool isOutputFinal () const

Determines if a node represents a final output.
• virtual bool isInput () const

Determines if a node represents an input.
• virtual bool isInputDomestic () const

Determines if a node represents a domestic input.
• virtual bool isInputImported () const

Determines if a node represents an imported input.
• virtual bool isInputFactor () const

Determines if a node represents a factor such as labor or capital.
• virtual bool isRegion () const

Determines if a node represents a region.
• virtual bool isCustomersUtility () const

Determines if a node represents customer utility (a kind of output).
• virtual bool isCustomersSavings () const

Determines if a node represents customer savings (a kind of input).
• virtual bool isCustomersProfits () const
  Determines if a node represents customer income from sectoral profits.
• virtual bool isCustomersProducts () const
  Determines if a node represents customer income from sectoral profits.
• virtual bool isCustomersIncome () const
  Determines if a node represents customer income from factors or investment profits.
• virtual bool isCustomersFactorsIncome () const
  Determines if a node represents customer income from factors.
• virtual bool isCustomersInvestmentIncome () const
  Determines if a node represents customer income from investment profits.
• virtual bool isCustomersPDTS () const
  Determines if a node represents customer abstract node "PDTS".
• virtual bool isAbstractOuputs () const
  Determines if a node represents an abstract entity.
• virtual bool isValidNode () const
  Determines if a node contains valid values.
• virtual void computeRho ()
  Compute rho = 1 - 1/sigma.
• virtual double computeAlphasSum ()
  Compute sum of alpha's in children.
• virtual void normalizeAlphas ()
  Normalize alpha's in children.
• virtual double ces (const bool=true)
  Compute the CES (Constant Elasticity of Substitution) value of this (output) entry.
• virtual CesNode & getParent () const
  Gets the parent node of this entry.
• virtual boost::shared_ptr<CesNode> getParentPtr () const
  Gets the smart pointer to the parent node of this entry.
• virtual void setParent (const CesNode &)
  Sets the parent node of this entry.
• virtual int getNodeNo () const
  Gets the node number of this entry.
• virtual void setNodeNo (const int)
  Sets the node number of this entry.
• virtual void computeAlphas ()
  Compute share parameter values in a production tree given expenses on inputs.
• virtual void computeExpense ()
  Compute expenses of non-input nodes from the expenses of input nodes.
• virtual bool isDomestic () const
4.1 CesNode Class Reference

- Determines if a node represents an input or an output that is domestic.
  - virtual void setDomestic (bool)
    Mark an input/output node as domestic.
- virtual List getVars (const string &, const string &)
  Collect a list of price, supply, and demand variable names from this node and its inputs.
- virtual string declareAmplVar (const string &)
  Returns a string that declares an AMPL variable.

Static Public Attributes

- static CesNode EMPTY_CESNODE = CesNode()
  An empty CesNode.

Friends

- std::ostream & operator<< (std::ostream &, CesNode &)
  Override << to print a CesNode instance values by depth-first search.

4.1.1 Detailed Description

A constant-elasticity-of-substitution node.

A constant-elasticity-of-substitution node that represents an output, or an input in a production tree. Alternatively, it could be used to model customer utility (output), savings (input), or a consumer product (input). Each node contains information such as constant elasticity of substitution, share parameter value, efficiency value, unit price, expense, tax, and whether it is domestic or not (imported).

Note

This class contains a (smart) pointer to a parent node and a vector of children nodes (if any).

4.1.2 Constructor & Destructor Documentation

4.1.2.1 CesNode::CesNode ( const int nodeNo = 0, const string class_label = string(),
  const string label = string(), const string comment = string() )

CesNode Constructor.
12 Class Documentation

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nodeNo</code></td>
<td>Integer index of this node. Default to 0.</td>
<td></td>
</tr>
<tr>
<td><code>class_label</code></td>
<td>Class label of this entry. Default to empty string.</td>
<td></td>
</tr>
<tr>
<td><code>label</code></td>
<td>String name of this entry. Default to empty string.</td>
<td></td>
</tr>
<tr>
<td><code>comment</code></td>
<td>A comment of this entry. Default to empty string.</td>
<td></td>
</tr>
</tbody>
</table>

4.1.2.2 CesNode::CesNode ( const int nodeNo, const string class_label, const string label, const string comment, const double sigma, const double alpha = 0, const double gamma = 1, const double amount = 0 )

CesNode Constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nodeNo</code></td>
<td>Integer index of this node. Default to 0.</td>
<td></td>
</tr>
<tr>
<td><code>class_label</code></td>
<td>Class label of this entry. Default to empty string.</td>
<td></td>
</tr>
<tr>
<td><code>label</code></td>
<td>String name of this entry. Default to empty string.</td>
<td></td>
</tr>
<tr>
<td><code>comment</code></td>
<td>A comment of this entry. Default to empty string.</td>
<td></td>
</tr>
<tr>
<td><code>alpha</code></td>
<td>Value of share parameter. Default to 0.</td>
<td></td>
</tr>
<tr>
<td><code>gamma</code></td>
<td>Value of efficiency parameter. Default to 1.</td>
<td></td>
</tr>
<tr>
<td><code>sigma</code></td>
<td>Value of elasticity of substitution. Default to 0.</td>
<td></td>
</tr>
<tr>
<td><code>amount</code></td>
<td>Amount of input. Default to 0.</td>
<td></td>
</tr>
</tbody>
</table>

4.1.3 Member Function Documentation

4.1.3.1 void CesNode::add ( const CesNode & cesNode ) [virtual]

Add a child node.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cesNode</td>
<td>A child node.</td>
</tr>
</tbody>
</table>

4.1.3.2 double CesNode::ces ( const bool childrenOnly = true ) [virtual]

Compute the CES (Constant Elasticity of Substitution) value of this (output) entry.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>childrenOnly</td>
<td>If true, computes CES value from children. Otherwise, computes CES value from all descendants. Default to true.</td>
</tr>
</tbody>
</table>
Returns

The CES value. Amount is set to the value for an output.

4.1.3.3 double CesNode::computeAlphasSum ( ) [virtual]

Compute sum of alpha’s in children.

Returns

Sum of alpha’s in children.

4.1.3.4 string CesNode::declareAmplVar ( const string & var ) [virtual]

Returns a string that declares an AMPL variable.

Parameters

| var | Name of the AMPL variable. |

Returns

A string AMPL statement that declares a variable unless the input string is empty.

4.1.3.5 CesNode & CesNode::get ( const size_t index ) [virtual]

Gets the child node by its index value.

Parameters

| index | Index of the child node. Value of index run from 0 to size()-1. |

Returns

The child node.

Reimplemented in PdnFcn.

4.1.3.6 CesNode & CesNode::get ( const string code, const bool childrenOnly = true ) [virtual]

Gets a descendant by its code.
Parameters

<table>
<thead>
<tr>
<th>code</th>
<th>Code of a descendant to look for in this tree.</th>
</tr>
</thead>
<tbody>
<tr>
<td>childrenOnly</td>
<td>True to conduct search in children only. False to conduct recursive search. Default to true.</td>
</tr>
</tbody>
</table>

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented in PdnFcn, and CustUtil.

4.1.3.7 CesNode & CesNode::get ( bool & found, const string code, const bool childrenOnly = true ) [virtual]

Gets a descendant by its code.

Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>found</th>
<th>True if the node with the input code is found. False otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>Code of a descendant to look for in this tree.</td>
<td></td>
</tr>
<tr>
<td>childrenOnly</td>
<td>True to conduct search in children only. False to conduct recursive search. Default to true.</td>
<td></td>
</tr>
</tbody>
</table>

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented in PdnFcn, and CustUtil.

4.1.3.8 double CesNode::getAlpha ( ) const [virtual]

Gets the share parameter of this (input) entry.

Returns

Share parameter of this (input) entry.

4.1.3.9 double CesNode::getAmount ( ) const [virtual]

Gets the amount of this (input/output) entry.
4.1 CesNode Class Reference

Returns

Amount of this (input/output) entry.

4.1.3.10 double CesNode::getAmountMax ( ) const [virtual]

Gets the maximum amount of this (input/output) entry.

Returns

Amount of this (input/output) entry.

4.1.3.11 string CesNode::getClassLabel ( ) const [virtual]

Gets the class label of this entry.

Returns

Class label of this entry.

4.1.3.12 string CesNode::getComment ( ) const [virtual]

Gets a comment of this entry.

Returns

A comment of this entry.

4.1.3.13 double CesNode::getExpense ( ) const [virtual]

Gets the expenditure associated with this entry.

Returns

Expenditure associated with this entry.

4.1.3.14 double CesNode::getGamma ( ) const [virtual]

Gets the efficiency parameter of this (input) entry.

Returns

Efficiency parameter of this (input) entry.
4.1.3.15 string CesNode::getLabel ( ) const [virtual]

Gets the label of this entry.

Returns
    Label of this entry.

4.1.3.16 int CesNode::getNodeNo ( ) const [virtual]

Gets the node number of this entry.

Returns
    The node number of this entry.

4.1.3.17 CesNode & CesNode::getParent ( ) const [virtual]

Gets the parent node of this entry.

Returns
    The parent node.

4.1.3.18 CesNodePtr CesNode::getParentPtr ( ) const [virtual]

Gets the smart pointer to the parent node of this entry.

Returns
    The smart pointer to the parent node.

4.1.3.19 double CesNode::getPrice ( ) const [virtual]

Gets the unit price of this entry.

Returns
    Unit price of this entry.
4.1.3.20 double CesNode::getRho ( ) const  [virtual]

Gets rho = 1 - 1/sigma where sigma is the elasticity of substitution of this entry.

Returns

    returns rho = 1 - 1/sigma.

4.1.3.21 double CesNode::getSigma ( ) const  [virtual]

Gets the elasticity of substitution of this entry.

Returns

    Elasticity of substitution of this entry.

4.1.3.22 size_t CesNode::getSize ( const bool childrenOnly = true )  [virtual]

Returns the number of children or grandchildren.

Parameters

    childrenOnly  If true, returns the number of children. Otherwise, returns the number
                  of grandchildren. Default to true.

Returns

    Number of children or grandchildren.

Reimplemented in PdnFcn, and CustUtil.

4.1.3.23 double CesNode::getTax ( ) const  [virtual]

Gets the tax of this entry.

Returns

    Tax of this entry.

4.1.3.24 List CesNode::getVars ( const string & finalOutput, const string & region )  [virtual]

Collect a list of price, supply, and demand variable names from this node and its inputs.
Parameters

| finalOutput | A string that represents the label of final output. |
| region      | Code of region associated with these entities. Default to empty string. |

Returns

A list of price, supply, and demand variable names with no repetition.

4.1.3.25 bool CesNode::isAbstractOutputs( ) const [virtual]

Determines if a node represents an abstract entity.

Returns

True if this node represents an abstract entity, otherwise false.

4.1.3.26 bool CesNode::isCustomersFactorsIncome( ) const [virtual]

Determines if a node represents customer income from factors.

Returns

True if this node represents customer utility from factors, otherwise false.

4.1.3.27 bool CesNode::isCustomersIncome( ) const [virtual]

Determines if a node represents customer income from factors or investment profits.

Returns

True if this node represents customer utility from factors or investment profits, otherwise false.

4.1.3.28 bool CesNode::isCustomersInvestmentIncome( ) const [virtual]

Determines if a node represents customer income from investment profits.

Returns

True if this node represents customer utility from investment profits, otherwise false.
4.1 CesNode Class Reference

4.1.3.29 bool CesNode::isCustomersPDTS() const [virtual]

Determines if a node represents customer abstract node "PDTS".

Returns

True if this node represents customer abstract node "PDTS", otherwise false.

4.1.3.30 bool CesNode::isCustomersProducts() const [virtual]

Determines if a node represents customer income from sectoral profits.

Returns

True if this node represents customer utility from sectoral profits, otherwise false.

4.1.3.31 bool CesNode::isCustomersProfits() const [virtual]

Determines if a node represents customer income from sectoral profits.

Returns

True if this node represents customer utility from sectoral profits, otherwise false.

4.1.3.32 bool CesNode::isCustomersSavings() const [virtual]

Determines if a node represents customer savings (a kind of input).

Returns

True if this node represents customer utility (a kind of input), otherwise false.

4.1.3.33 bool CesNode::isCustomersUtility() const [virtual]

Determines if a node represents customer utility (a kind of output).

Returns

True if this node represents customer utility (a kind of output), otherwise false.
4.1.3.34 bool CesNode::isDomestic ( ) const [virtual]

Determines if a node represents an input or an output that is domestic.

Returns

True if this node represents an input or an output that is domestic, otherwise false.

4.1.3.35 bool CesNode::isInput ( ) const [virtual]

Determines if a node represents an input.

Returns

True if this node represents an input (domestic or imported), otherwise false.

4.1.3.36 bool CesNode::isInputDomestic ( ) const [virtual]

Determines if a node represents a domestic input.

Returns

True if this node represents a domestic input, otherwise false.

4.1.3.37 bool CesNode::isInputFactor ( ) const [virtual]

Determines if a node represents a factor such as labor or capital.

Returns

True if this node represents a factor, otherwise false.

4.1.3.38 bool CesNode::isInputImported ( ) const [virtual]

Determines if a node represents an imported input.

Returns

True if this node represents an imported input, otherwise false.
4.1 CesNode Class Reference

4.1.3.39 bool CesNode::isOutput() const [virtual]
Determines if a node represents an output.

Returns
True if this node represents an output (final or immediate), otherwise false.

4.1.3.40 bool CesNode::isOutputFinal() const [virtual]
Determines if a node represents a final output.

Returns
True if this node represents a final output, otherwise false.

4.1.3.41 bool CesNode::isOutputImmediate() const [virtual]
Determines if a node represents an immediate output.

Returns
True if this node represents an immediate output, otherwise false.

4.1.3.42 bool CesNode::isRegion() const [virtual]
Determines if a node represents a region.

Returns
True if this node represents a region, otherwise false.

4.1.3.43 bool CesNode::isValidNode() const [virtual]
Determines if a node contains valid values.

Returns
True if this node contains valid values, otherwise false.
4.1.3.44 void CesNode::normalizeAlphas () [virtual]

Normalize alpha’s in children.

Returns
Recompute and set alpha’s in children so that they sum to one.

4.1.3.45 bool CesNode::operator!= ( CesNode & cesNode ) [virtual]

Overload operator != for comparing two CesNode instances.

Parameters

| cesNode | Right-hand-side CesNode. |

Returns
False if this CesNode and cesNode have the same node no, class label, label, alpha, gamma, sigma, amount, and number of children.

4.1.3.46 bool CesNode::operator== ( CesNode & cesNode ) [virtual]

Overload operator == for comparing two CesNodes (non-recursively).

Parameters

| cesNode | Right-hand-side CesNode instance. |

Returns
True if this CesNode instance and CesNode have the same node no, class label, label, alpha, gamma, sigma, amount, and number of children.

4.1.3.47 void CesNode::print ( std::ostream & os ) [virtual]

Recursively print node numbers of a node and its descendants to standard output.

Parameters

| os | An ostream instance. |
4.1 CesNode Class Reference

4.1.3.48  void CesNode::printNode() [virtual]
Print this node to standard output.

4.1.3.49  std::string CesNode::printstr(CesNode & root) [virtual]
Recursively print node numbers of a node and its descendants to a string.

Parameters

| root | A CesNode instance. |

Returns
A string with node numbers of a node and its descendants in nested parentheses.

4.1.3.50  void CesNode::setAlpha(const double alpha) [virtual]
Sets the share parameter of this (input) entry.

Parameters

| alpha | Share parameter of this (input) entry. Between 0 and 1 inclusive. |

4.1.3.51  void CesNode::setAmount(const double amount) [virtual]
Sets the amount of this (input/output) entry.

Parameters

| amount | Amount of this (input/output) entry. Non-negative. |

4.1.3.52  void CesNode::setAmountMax(const double amountMax) [virtual]
Sets the maximum amount of this (input/output) entry.

Parameters

| amountMax | Amount of this (input/output) entry. Non-negative. |
### 4.1.3.53 void CesNode::setClassLabel ( const string class_label ) [virtual]

Sets the class label of this entry.

**Parameters**

| class_label | Class label of this entry. |

### 4.1.3.54 void CesNode::setComment ( const string comment ) [virtual]

Sets a comment of this entry.

**Parameters**

| comment | A comment of this entry. |

### 4.1.3.55 void CesNode::setDomestic ( bool domestic ) [virtual]

Mark an input/output node as domestic.

**Parameters**

| domestic | Set to true to mark this input/output node as domestic, false otherwise. |

### 4.1.3.56 void CesNode::setExpense ( const double expense ) [virtual]

Sets the expenditure of this (input/output) entry.

**Parameters**

| expense | Expenditure of this (input/output) entry. Non-negative. |

### 4.1.3.57 void CesNode::setGamma ( const double gamma ) [virtual]

Sets the efficiency parameter of this (input) entry.

**Parameters**

| gamma | Efficiency parameter of this (input) entry. Between 0 and 1 inclusive. |
4.1 CesNode Class Reference

4.1.3.58 void CesNode::setLabel ( const string label ) [virtual]
Sets label of this entry.

Parameters

| label | Label of this entry. |

4.1.3.59 void CesNode::setParent ( const CesNode & parent ) [virtual]
Sets the parent node of this entry.

Parameters

| parent | Parent node of this entry. |

4.1.3.60 void CesNode::setPrice ( const double price ) [virtual]
Sets the price of this (input/output) entry.

Parameters

| price | Price of this (input/output) entry. Non-negative. |

4.1.3.61 void CesNode::setSigma ( const double sigma, const bool childrenOnly = true ) [virtual]
Sets the elasticity of substitution of this (input) entry. It also updates rho.

Parameters

| sigma | Elasticity of substitution of this (input) entry. Non-negative. To set sigma to positive infinity, use util::INF. |
| childrenOnly | If true, returns the number of children. Otherwise, returns the number of grandchildren. Default to true. |

Reimplemented in PdnFcn.

4.1.3.62 void CesNode::setTax ( const double tax ) [virtual]
Sets the tax of this (input/final output) entry.
26 Class Documentation

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tax</code></td>
<td>Tax of this (input/final output) entry. If negative, it is subsidy.</td>
</tr>
</tbody>
</table>

4.1.4 Friends And Related Function Documentation

4.1.4.1 `std::ostream& operator<<( std::ostream & os, CesNode & root )` [friend]

Override `<<` to print a `CesNode` instance values by depth-first search.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>os</code></td>
<td>ostream Default is standard output.</td>
</tr>
<tr>
<td><code>root</code></td>
<td>A <code>CesNode</code> instance.</td>
</tr>
</tbody>
</table>

Returns

An ostream instance.

The documentation for this class was generated from the following files:

• CesNode.h
• CesNode.cpp

4.2 Cgds Class Reference

A number-indexed list of cgds.

#include <Cgds.h>

Inheritance diagram for Cgds:

```
List

Cgds
```

Public Member Functions

• `Cgds()`  
  
  `Cgds` default constructor.
• `Cgds(const string)`
4.3 Cge Class Reference

*Cgds* constructor reads an external standardized regions file.

- virtual ~*Cgds* ()

  *Cgds default destructor.*

4.2.1 Detailed Description

A number-indexed list of cgds.

A numbered list of codes that represent cgds.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 *Cgds::Cgds* (const string *filename*)

*Cgds* constructor reads an external standardized regions file.

Parameters

- *filename* path to a regions file.

The documentation for this class was generated from the following files:

- Cgds.h
- Cgds.cpp

4.3 Cge Class Reference

Class for constructing a CGE (computational general equilibrium) problem.

#include <Cge.h>

Public Member Functions

- *Cge* ()

  *Cge default constructor.*

- *Cge* (const string &, const string &, const string &)

  *Cge* constructor for reading input files of standardized production function file, customer utility function file, and social accounting matrices.

- *Cge* (*PdnFcn* &, *CustUtil* &)

  *Cge* constructor for reading input files of standardized production function file and customer utility function file.
• **Cge** *(PdnFcn &, CustUtil &, Sams &)*

  *Cge* constructor for reading input files of standardized production function file, customer utility function file, and social accounting matrices.

• virtual \(~\text{Cge} ()\)

  *Cge* default destructor.

• virtual void **write** (const string)

  Writes an external standardized production function file.

• virtual void **extractTaxes** (const Sam &, const string=string())

  Extract taxes into a product tree's inputs (leaves) from a user-input *Sam* matrix.

• virtual void **extractExpenses** (const Sam &, const string=string())

  Extract expenses into a product tree's inputs (leaves) from a user-input *Sam* matrix.

• virtual void **estimateAlphas** (const Sam &, const string=string())

  Estimates alphas of a product tree from a user-input *Sam* matrix.

• virtual string **printCge** (const string=string("cimearth"))

  Prints the CGE problem statement of an output to a string.

• virtual string **printMcp** ()

  Prints the MCP form of a CGE problem statement of an output to a string.

• virtual void **writeMcp** (string=string("cimearth.ampl"), string=string("cimearth.-log"))

  Writes the MCP form of a CGE problem statement of an output to an external file.

• virtual **PdnFcn &** getPdnFcn ()

  Gets a production function associated with this CGE model.

• virtual **CustUtil &** getCustUtil ()

  Gets a customer utility function associated with this CGE model.

• virtual **Sams &** getSams ()

  Gets social accounting matrices associated with this CGE model.

### 4.3.1 Detailed Description

Class for constructing a CGE (computational general equilibrium) problem.

Class for reading input files of standardized production function, customer utility functions, and social accounting matrices.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 Cge::Cge ( const string & pdnFcnPath, const string & custUtilPath, const string & samsPath )

*Cge* constructor for reading input files of standardized production function file, customer utility function file, and social accounting matrices.
4.3 Cge Class Reference

Parameters

<table>
<thead>
<tr>
<th>pdnFcnPath</th>
<th>Path to a production-function specification file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>custUtilPath</td>
<td>Path to a customer-utility specification file.</td>
</tr>
<tr>
<td>samsPath</td>
<td>Path to a directory with SAM files.</td>
</tr>
</tbody>
</table>

4.3.2.2 Cge::Cge ( PdnFcn & pdnFcnIn, CustUtil & custUtilIn )

Cge constructor for reading input files of standardized production function file and customer utility function file.

Parameters

<table>
<thead>
<tr>
<th>pdnFcnIn</th>
<th>Path to a production-function specification file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>custUtilIn</td>
<td>Path to a customer-utility specification file.</td>
</tr>
</tbody>
</table>

4.3.2.3 Cge::Cge ( PdnFcn & pdnFcnIn, CustUtil & custUtilIn, Sams & samsIn )

Cge constructor for reading input files of standardized production function file, customer utility function file, and social accounting matrices.

Parameters

<table>
<thead>
<tr>
<th>pdnFcnIn</th>
<th>Path to a production-function specification file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>custUtilIn</td>
<td>Path to a customer-utility specification file.</td>
</tr>
<tr>
<td>samsIn</td>
<td>Path to a directory with SAM files.</td>
</tr>
</tbody>
</table>

4.3.3 Member Function Documentation

4.3.3.1 void Cge::estimateAlphas ( const Sam & sam, const string output = string() ) [virtual]

Estimates alphas of a product tree from a user-input Sam matrix.

Parameters

<table>
<thead>
<tr>
<th>sam</th>
<th>A Sam matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Label of targeted production function whose alphas are to be estimated. Optional. Default to empty string, in which case the method will attempt to estimate alphas of all leave nodes in all the production trees associated with the region given by the SAM.</td>
</tr>
</tbody>
</table>
4.3.3.2 void Cge::extractExpenses ( const Sam & sam, const string output = string() ) [virtual]

Extract expenses into a product tree's inputs (leaves) from a user-input Sam matrix.

Parameters

<table>
<thead>
<tr>
<th>sam</th>
<th>A SAM matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Label of targeted production function whose alphas are to be estimated.</td>
</tr>
</tbody>
</table>

4.3.3.3 void Cge::extractTaxes ( const Sam & sam, const string output = string() ) [virtual]

Extract taxes into a product tree's inputs (leaves) from a user-input Sam matrix.

Parameters

<table>
<thead>
<tr>
<th>sam</th>
<th>A SAM matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Label of targeted production function whose taxes are to be extracted.</td>
</tr>
</tbody>
</table>

4.3.3.4 CustUtil & Cge::getCustUtil ( ) [virtual]

Gets a customer utility function associated with this CGE model.

Returns

A CustUtil instance of this CGE model.

Note

If the returned instance is modified, then this CGE instance is also modified.

4.3.3.5 PdnFcn & Cge::getPdnFcn ( ) [virtual]

Gets a production function associated with this CGE model.

Returns

A PdnFcn instance of this CGE model.

Note

If the returned instance is modified, then this CGE instance is also modified.
4.3 Cge Class Reference

4.3.3.6 Sams & Cge::getSams ( ) [virtual]

Gets social accounting matrices associated with this CGE model.

Returns

A Sams instance of this CGE model.

Note

If the returned instance is modified, then this CGE instance is also modified.

4.3.3.7 string Cge::printCge ( const string \textit{filename} = string("cimearth") ) [virtual]

Prints the CGE problem statement of an output to a string.

Parameters

\begin{tabular}{|l|p{30cm}|}
\hline
\textbf{filename} & path to the SAMS file. If the file already existed before, its previous content is deleted and replaced by the new one. \\
\hline
\end{tabular}

Returns

string A string expression of the CGE problem statement.

4.3.3.8 string Cge::printMcp ( ) [virtual]

Prints the MCP form of a CGE problem statement of an output to a string.

Returns

string A string expression of the CGE problem statement.

4.3.3.9 void Cge::write ( const string \textit{filename} ) [virtual]

Writes an external standardized production function file.

Parameters

\begin{tabular}{|l|p{30cm}|}
\hline
\textbf{filename} & path to the production function file. If the file already existed before, its previous content is deleted and replaced by the new one. \\
\hline
\end{tabular}
4.3.3.10 void Cge::writeMcp ( string amplFile = string("cimearth.ampl"),
                string logFile = string("cimearth.log") ) [virtual]

Writes the MCP form of a CGE problem statement of an output to an external file.

Parameters

<table>
<thead>
<tr>
<th>amplFile</th>
<th>path to the SAMS file. If the file already existed before, its previous content is deleted and replaced by the new one. Default to &quot;cimearth.-log&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>logFile</td>
<td>AMPL log file name. Default to &quot;cimearth.ampl&quot;.</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following files:

- Cge.h
- Cge.cpp

4.4 CustUtil Class Reference

Class for constructing a customer-utility tree.

#include <CustUtil.h>

Inheritance diagram for CustUtil:

```
CesNode
   |
   V
CustUtil
```

Public Member Functions

- **CustUtil ()**
  
  *CustUtil* default constructor. Create a child node USA region and a grandchild node USA customers by default.

- **CustUtil (const string &)**
  
  *CustUtil* constructor that reads an external customer utility file.

- **CustUtil (const Sam &)**
  
  *CustUtil* constructor that reads an external standardized SAM.

- **CustUtil (PdnFcn &)**
  
  *CustUtil* constructor that constructs sectors from a production function.
• virtual ~CustUtil ()
  
  \textit{CustUtil} default destructor.
• virtual void \texttt{print()} ()
  
  Print this production function.
• virtual string \texttt{printCge() (CesNode &)}
  
  \textit{Prints the CGE problem statement of an output to a string.}
• virtual string \texttt{printCge() ()}
  
  \textit{Prints the CGE problem statement of all outputs.}
• virtual string \texttt{printMcp() (string &varlists)}
  
  \textit{Prints the MCP problem statement of all outputs.}
• virtual string \texttt{printObjective() (CesNode &)}
  
  \textit{Prints the objective of maximizing customer utility in a CGE problem.}
• virtual string \texttt{printConstraints() (CesNode &)}
  
  \textit{Print the CES constraints of an output and the non-negative constraints of its inputs.}
• virtual string \texttt{printFocCesConstraints() (string &., const string &.=string("USA"))}
  
  \textit{Print the CES constraints of a customer utility output.}
• virtual string \texttt{printFocConstraints() (CesNode &., const string &., const string &.)}
  
  \textit{Print the CES constraints of an output and the non-negative constraints of its inputs.}
• virtual string \texttt{printRevenue() (const CesNode &)}
  
  \textit{Print the after-tax revenue of an output (sector).}
• virtual string \texttt{printExpense() (CesNode &)}
  
  \textit{Print the after-tax expense of an input.}
• virtual string \texttt{printCesConstraints() (CesNode &)}
  
  \textit{Print the CES constraints of a customer utility output.}
• string \texttt{printFocIncome() (const string &.=string("USA"))}
  
  \textit{Print the income of a customer utility output.}
• string \texttt{printFocExpenses() (const string &.=string("USA"))}
  
  \textit{Print the expense of a customer utility output.}
• virtual string \texttt{printIncomeExpenseConstraints() (CesNode &)}
  
  \textit{Print the income constraints of a customer utility output.}
• virtual string \texttt{printFocIncomeExpenseConstraint() (string &., const string &.=string("USA"))}
  
  \textit{Print the income constraints of a customer utility output.}
• virtual string \texttt{printFocNonNegativeConstraint() (string &., const string &.=string("USA"))}
  
  \textit{Print the complementarity conditions related to variables of a customer utility output.}
• virtual string \texttt{printNonNegativeConstraint() (CesNode &)}
  
  \textit{Print the non-negative constraint of an input.}
• virtual string \texttt{printFocNonNegativeConstraint() (CesNode &., const string, const string, string &.)}
Print the non-negative constraint of an input.

- virtual void read (const string)
  
  Reads an external standardized utility function.
- virtual void extractCesNode (string, CesNode &)
  
  Extract a CES node from a string in standardized format.
- virtual void computeAlphas ()
  
  Compute share parameter values in a production tree given expenses on inputs.
- virtual CesNode & get (const string, const bool=true)
  
  Gets a descendant by its code.
- virtual CesNode & get (bool &, const string, const bool=true)
  
  Gets a descendant by its code.
- virtual size_t getSize (const bool=true)
  
  Returns the number of children or grandchildren.
- virtual CustUtil & operator= (CustUtil &)
- Factors getFactors () const
  
  Return factors.
- void setFactors (Factors factors)
  
  Set factors.
- Sectors getSectors () const
  
  Return sectors.
- void setSectors (Sectors sectors)
  
  Set sectors.

4.4.1 Detailed Description

Class for constructing a customer-utility tree.

Class for constructing a customer-utility tree using the class CesNode.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CustUtil::CustUtil ( const string & filename )

CustUtil constructor that reads an external customer utility file.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Path to an external customer utility file.</td>
</tr>
</tbody>
</table>
4.4 CustUtil Class Reference

4.4.2.2 CustUtil::CustUtil (const Sam & sam)

CustUtil constructor that reads an external standardized SAM.

Parameters

| sam | A SAM instance. |

4.4.2.3 CustUtil::CustUtil (PdnFcn & pdnFcn)

CustUtil constructor that constructs sectors from a production function.

Parameters

| pdnFcn | A production function instance. |

4.4.3 Member Function Documentation

4.4.3.1 void CustUtil::extractCesNode (string line, CesNode & node) [virtual]

Extract a CES node from a string in standardized format.

Parameters

| line | String in standardized format of a CES node. |
| node | A CesNode for storing extracted values from input string. |

4.4.3.2 CesNode & CustUtil::get (const string code, const bool childrenOnly = true) [virtual]

Gets a descendant by its code.

Parameters

| code | Code of a descendant to look for in this tree. |
| childrenOnly | True to conduct search in children only. False to conduct recursive search. Default to true. |

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented from CesNode.
4.4.3.3 CesNode & CustUtil::get ( bool & found, const string code, const bool childrenOnly = true ) [virtual]

Gets a descendant by its code.

Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>found</th>
<th>True if the node with the input code is found. False otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td></td>
<td>Code of a descendant to look for in this tree.</td>
</tr>
<tr>
<td>childrenOnly</td>
<td></td>
<td>True to conduct search in children only. False to conduct recursive search. Default to true.</td>
</tr>
</tbody>
</table>

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented from CesNode.

4.4.3.4 size_t CustUtil::getSize ( const bool childrenOnly = true ) [virtual]

Returns the number of children or grandchildren.

Parameters

| childrenOnly | If true, returns the number of children. Otherwise, returns the number of grandchildren. Default to true. |

Returns

Number of children or grandchildren.

Reimplemented from CesNode.

4.4.3.5 CustUtil & CustUtil::operator= ( CustUtil & pdfFcn ) [virtual]

Parameters

| pdfFcn | Production Function to be assigned to this instance. |

4.4.3.6 string CustUtil::printCesConstraints ( CesNode & node ) [virtual]

Print the CES constraints of a customer utility output.
Returns

A string expression of the customer utility’s CES constraints.

4.4.3.7  string CustUtil::printCge ( CesNode & output ) [virtual]

Prints the CGE problem statement of an output to a string.

Parameters

| output | A final output. |

Returns

string A string expression of the CGE problem statement.

4.4.3.8  string CustUtil::printCge ( ) [virtual]

Prints the CGE problem statement of all outputs.

Returns

string A string expression of the CGE problem statement.

4.4.3.9  string CustUtil::printConstraints ( CesNode & output ) [virtual]

Print the CES constraints of an output and the non-negative constraints of its inputs.

Returns

A string expression of the constraints.

4.4.3.10  string CustUtil::printExpense ( CesNode & node ) [virtual]

Print the after-tax expense of an input.

Returns

A string expression of the after-tax expense.
4.4.3.11 string CustUtil::printFocCesConstraints ( string & varlist, const string & regionLabel = string("USA") ) [virtual]

Print the CES constraints of a customer utility output.

Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>varlist</th>
<th>AMPL declaration of variable(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>regionLabel</td>
<td>string of Region Code. Default to &quot;USA&quot;.</td>
</tr>
</tbody>
</table>

Returns

A string expression of the customer utility's CES constraints.

4.4.3.12 string CustUtil::printFocConstraints ( CesNode & output, const string & regionLabel, const string & outputLabel ) [virtual]

Print the CES constraints of an output and the non-negative constraints of its inputs.

Returns

A string expression of the constraints.

4.4.3.13 string CustUtil::printFocExpenses ( const string & regionLabel = string("USA") )

Print the expense of a customer utility output.

Parameters

| regionLabel | Region code. Default to "USA". |

Returns

A string expression of the customer utility’s income constraints.

4.4.3.14 string CustUtil::printFocIncome ( const string & regionLabel = string("USA") )

Print the income of a customer utility output.
4.4 CustUtil Class Reference

Parameters

| regionLabel | Region code. Default to "USA". |

Returns

A string expression of the customer utility's income constraints.

4.4.3.15 string CustUtil::printFocIncomeExpenseConstraint ( string & varlist, const string & regionLabel = string("USA") ) [virtual]

Print the income constraints of a customer utility output.

Parameters

| out varlist | AMPL declaration of variable(s). |
| in regionLabel | Region code. Default to "USA". |

Returns

A string expression of the customer utility's income constraints.

4.4.3.16 string CustUtil::printFocNonNegativeConstraint ( string & varlist, const string & regionLabel = string("USA") ) [virtual]

Print the complementarity conditions related to variables of a customer utility output.

Parameters

| out varlist | AMPL declaration of variable(s). |
| in regionLabel | Region code. Default to "USA". |

Returns

A string expression of the customer utility's complementarity conditions.

4.4.3.17 string CustUtil::printFocNonNegativeConstraint ( CesNode & node, const string regionLabel, const string outputLabel, string & varlist ) [virtual]

Print the non-negative constraint of an input.

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
Returns

A string expression of the non-negative constraint of an input.

4.4.3.18  string CustUtil::printIncomeExpenseConstraints ( CesNode & node )  [virtual]

Print the income constraints of a customer utility output.

Returns

A string expression of the customer utility’s income constraints.

4.4.3.19  string CustUtil::printMcp ( string & varlists )  [virtual]

Prints the MCP problem statement of all outputs.

Returns

string A string expression of the CGE problem statement.

4.4.3.20  string CustUtil::printNonNegativeConstraint ( CesNode & node )  [virtual]

Print the non-negative constraint of an input.

Returns

A string expression of the non-negative constraint of an input.

4.4.3.21  string CustUtil::printObjective ( CesNode & output )  [virtual]

Prints the objective of maximizing customer utility in a CGE problem.

Returns

string A string expression of the objective of maximizing customer utility in a CGE problem.
4.4 CustUtil Class Reference

4.4.3.22  string CustUtil::printRevenue ( const CesNode & output ) [virtual]

Print the after-tax revenue of an output (sector).

Returns
   A string expression of the after-tax revenue.

4.4.3.23  void CustUtil::read ( const string filename ) [virtual]

Reads an external standardized utility function.

Parameters

| filename | Path to a utility-function specification file. Note that it is case sensitive. Preconditions: comma-separated file with first line being and first column being headers, followed by lines and columns of non-negative values, and last row and last column holding the total sum of the column values and row values respectively. |

4.4.3.24  void CustUtil::setFactors ( Factors factors )

Set factors.

Parameters

| factors | Set factors to this input value. |

4.4.3.25  void CustUtil::setSectors ( Sectors sectors )

Set sectors.

Parameters

| sectors | Set sectors to this input value. |

The documentation for this class was generated from the following files:

- CustUtil.h
- CustUtil.cpp

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
4.5 Factors Class Reference

A number-indexed list of factors.

#include <Factors.h>

Inheritance diagram for Factors:

```
List

Factors
```

Public Member Functions

- **Factors ()**
  
  *Factors default constructor.

- **Factors (const string)**
  
  *Factors constructor reads an external standardized regions file.

- **virtual ~Factors ()**
  
  *Factors default destructor.

4.5.1 Detailed Description

A number-indexed list of factors.

A numbered list of codes that represent factors.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Factors::Factors (const string *filename*)

*Factors* constructor reads an external standardized regions file.

**Parameters**

| `filename` | path to a regions file. |

The documentation for this class was generated from the following files:

- Factors.h
- Factors.cpp
4.6 List Class Reference

Parent class for a numbered list of codes.

```cpp
#include <List.h>
```

Inheritance diagram for List:

![Inheritance Diagram](image)

Public Member Functions

- `List ()`
  - `List default constructor.`
- `List (const string)`
  - `List Constructor reads an external standardized List file.`
- `virtual ~List ()`
  - `List default destructor.`
- `virtual void write (const string) const`
  - `Writes an external standardized List file.`
- `virtual size_t getSize () const`
  - `Gets the length of a list.`
- `virtual string getCode (const size_t) const`
  - `Returns the code by its index, or an empty string if no such index exists.`
- `virtual int getIndex (const string) const`
  - `Returns an index of positive integer by its code, or -1 if no such code exists.`
- `virtual UnsignedInt_Vector getIndices () const`
  - `Returns a vector of positive integers in this list.`
- `virtual void setCode (const string)`
  - `Adds a code to the list if the code is not yet in the list.`
- `virtual void replaceCode (const string, const string)`
  - `Replace an existing code with a new one. If the new code already exists before replacement, the old code is erased instead.`
- `virtual void read (const string)`
  - `Reads an external standardized List file.`
- `virtual void print (int=-1) const`
  - `Prints the values of this List instance.`
4.6.1 Detailed Description

Parent class for a numbered list of codes.

Parent class for a numbered list of codes that represent sectors, factors, regions, or margins.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 List::List ( const string filename )

List Constructor reads an external standardized List file.

Parameters

filename | Path to a List file.

4.6.3 Member Function Documentation

4.6.3.1 void List::add ( const List & list ) [virtual]

Add a list to this list.

Parameters

list | A list to be merged to this list.

4.6.3.2 bool List::codeExist ( const string code ) const [virtual]

Returns true if the code exists, or false if no such code exists.
### 4.6 List Class Reference

#### Parameters

<table>
<thead>
<tr>
<th>code</th>
<th>Code.</th>
</tr>
</thead>
</table>

#### 4.6.3.3 void List::erase ( const string code, const bool ifReindex = false ) [virtual]

Remove a code from the list if the code is already in the list.

<table>
<thead>
<tr>
<th>code</th>
<th>Code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifReindex</td>
<td>Default to false. Set it to true to reindex the indices in this list to 1,2,...</td>
</tr>
</tbody>
</table>

#### 4.6.3.4 void List::erase ( const size_t index, const bool ifReindex = false ) [virtual]

Remove a code by its index key from the list if the code is already in the list.

<table>
<thead>
<tr>
<th>index</th>
<th>Index of the code to be removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifReindex</td>
<td>Default to false. Set it to true to reindex the indices in this list to 1,2,...</td>
</tr>
</tbody>
</table>

#### 4.6.3.5 string List::getCode ( const size_t index ) const [virtual]

Returns the code by its index, or an empty string if no such index exists.

| index | Positive integer. Index of a code. |

#### 4.6.3.6 int List::getIndex ( const string code ) const [virtual]

Returns an index of positive integer by its code, or -1 if no such code exists.

| code | Code associated with the index. |
4.6.3.7 size_t List::getSize() const [virtual]

Gets the length of a list.

Returns

Length of a list.

4.6.3.8 void List::print ( int n = -1 ) const [virtual]

Prints the values of this List instance.

Parameters

| n | Default value -1 means printing all list values. Any positive integer value less than list size means printing the first n list codes. |

4.6.3.9 void List::read ( const string filename ) [virtual]

Reads an external standardized List file.

Parameters

| filename | Path to a file that has a header, and two columns, which are separated by a column of ','. The first column contains integers. The second column contains strings. |

4.6.3.10 void List::replaceCode ( const string oldCode, const string newCode ) [virtual]

Replace an existing code with a new one. If the new code already exists before replacement, the old code is erased instead.

Parameters

| oldCode | Old code string to be replaced. |
| newCode | New code string to replace old code string. |

4.6.3.11 void List::setCode ( const string code ) [virtual]

Adds a code to the list if the code is not yet in the list.
4.7 Mapper Class Reference

A collection of tuples from the class MapperEntry.

#include <Mapper.h>

Public Member Functions

- Mapper ()
  
  Mapper default constructor.
- Mapper (const string)
  
  Mapper Constructor reads an external standardized Mapper file.
- virtual ~Mapper ()
  
  Mapper default destructor.
- virtual void setSize (const size_t)
  
  Sets the length of a mapper.
- virtual void read (const string)
  
  Reads an external standardized Mapper file.
- virtual void setMapperEntry (const MapperEntry &)
  
  Sets (or adds) a mapper entry.
- virtual void write (const string) const
  
  Writes an external standardized Mapper file.
- virtual size_t getSize () const
Gets the length of a mapper.

- virtual **Mapper** getOutCodes (const string) const

  Returns the output code(s) given its optional input code.

- virtual **Mapper** getInCodes (const string) const

  Returns a list of input codes that are associated with an output code.

- virtual **List** getInCodesList (const string=string()) const

  Returns a list of input codes.

- virtual **List** getOutCodesList (const string=string()) const

  Returns a list of output codes.

- virtual string getPath () const

  Returns the path of this mapper.

- virtual void setPath (const string=string())

  Sets the path of this mapper.

- virtual **MapperEntry** getMapperEntry (const int) const

  Gets a mapper entry by index.

- virtual **Double_Sparse_Matrix** getQ () const

  Returns a dis/aggregation matrix.

- virtual void computeQ ()

  Computes a dis/aggregation matrix.

- virtual void setQ (const **Double_Sparse_Matrix**)

  Sets a dis/aggregation matrix.

- virtual bool isValidAgg () const

  Checks if this mapper is a valid aggregation map.

- virtual bool isValidDisagg () const

  Checks if this mapper is a valid disaggregation map.

- virtual void print () const

  Prints this mapper entry.

**Protected Member Functions**

- virtual int inCodeOutCodeExist (const string, const string) const

  Checks if a pair of input codes and output codes already exist.

### 4.7.1 Detailed Description

A collection of tuples from the class **MapperEntry**.

Tuples of in-codes, out-codes, and weights. Multiple in-code values can be aggregated into one out-code value, or an in-code value could be disaggregated into multiple out-code values, with numerical weights specified in either case.
4.7 Mapper Class Reference

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Mapper::Mapper ( const string filename )

Mapper Constructor reads an external standardized Mapper file.

Parameters

| filename | Path to a file. Note that it is case sensitive. In the mapper, there is no header, and it has two or three columns, which are separated by a column of ','. first column contains input codes (strings). The second column contains output codes (strings). The third column is optional, and if exists contains numerical weights of non-negative values. |

Note

If the input mapper file contains more than one entry of input-output codes of the same value, a Mapper instance stores only the latest entry.

4.7.3 Member Function Documentation

4.7.3.1 Mapper Mapper::getInCodes ( const string outCode ) const [virtual]

Returns a list of input codes that are associated with an output code.

Parameters

| outCode | Code of an output. |

4.7.3.2 List Mapper::getInCodesList ( const string outCode = string() ) const [virtual]

Returns a list of input codes.

Parameters

| outCode | Code of an output. OPTIONAL. Default to empty string. |

Returns

A list of input codes corresponding to outCode. If outCode is not provided, we return a list of all input codes.
4.7.3.3  MapperEntry Mapper::getMapperEntry ( const int index ) const  [virtual]

Gets a mapper entry by index.

Parameters

index  Positive Integer. Index of a mapper entry.

Returns

A MapperEntry instance, or an empty MapperEntry instance if no such index value exist.

4.7.3.4  Mapper Mapper::getOutCodes ( const string inCode ) const  [virtual]

Returns the output code(s) given its optional input code.

Parameters

inCode  input code(s). Code of an input.

4.7.3.5  List Mapper::getOutCodesList ( const string inCode = string() ) const  [virtual]

Returns a list of output codes.

Parameters

inCode  Code of an input. OPTIONAL. Default to empty string.

4.7.3.6  size_t Mapper::getSize ( ) const  [virtual]

Gets the length of a mapper.

Returns

Length of a mapper.

4.7.3.7  int Mapper::inCodeOutCodeExist ( const string inCode, const string outCode ) const  [protected, virtual]

Checks if a pair of input codes and output codes already exist.
4.7 Mapper Class Reference

Parameters

<table>
<thead>
<tr>
<th>inCode</th>
<th>Code of an input.</th>
</tr>
</thead>
<tbody>
<tr>
<td>outCode</td>
<td>Code of an output.</td>
</tr>
</tbody>
</table>

Returns

0 if the pair do not exist, or positive integer that indicates the index position.

4.7.3.8 bool Mapper::isValidAgg ( ) const [virtual]

Checks if this mapper is a valid aggregation map.

Returns

True if this mapper instance is a valid aggregation map, false otherwise.

4.7.3.9 bool Mapper::isValidDisagg ( ) const [virtual]

Checks if this mapper is a valid disaggregation map.

Returns

True if this mapper instance is a valid disaggregation map, false otherwise.

4.7.3.10 void Mapper::read ( const string filename ) [virtual]

Reads an external standardized Mapper file.

Parameters

| filename | Path to a file that has no header, and two or three columns, which are separated by a column of ‘,’. The first column contains input codes (strings). The second column contains output codes (strings). The third column is optional, and if exists contains numerical weights of non-negative values. |

Note

If the input mapper file contains more than one entry of input-output codes of the same value, a Mapper instance stores only the latest entry.
4.7.3.11 void Mapper::setMapperEntry ( const MapperEntry & mapperEntry ) [virtual]

Sets (or adds) a mapper entry.

Parameters

| mapperEntry | A MapperEntry instance. |

4.7.3.12 void Mapper::setPath ( const string path = string() ) [virtual]

Sets the path of this mapper.

Parameters

| path | Path of this mapper. |

4.7.3.13 void Mapper::setQ ( const Double_Sparse_Matrix P ) [virtual]

Sets a dis/aggregation matrix.

Parameters

| P | A double sparse matrix. |

4.7.3.14 void Mapper::setSize ( const size_t n ) [virtual]

Sets the length of a mapper.

Parameters

| n | Positive Integer. Length of a mapper. |

4.7.3.15 void Mapper::write ( const string filename ) const [virtual]

Writes an external standardized Mapper file.

Parameters

| filename | Path to a Mapper file. If the file already existed before, its previous content is deleted and replaced by the new one. |
The documentation for this class was generated from the following files:

- Mapper.h
- Mapper.cpp

4.8 MapperEntry Class Reference

A tuple of in-code, out-code, and weight.

```cpp
#include <MapperEntry.h>
```

Public Member Functions

- `MapperEntry ()`
  
  MapperEntry default constructor.

- `MapperEntry (const string, const string, const double=1.0)`
  
  MapperEntry Constructor.

- `virtual ~MapperEntry ()`
  
  MapperEntry default destructor.

- `virtual string getInCode () const`
  
  Gets the input-region code of this entry.

- `virtual string getOutCode () const`
  
  Gets the output-region code of this entry.

- `virtual double getWeight () const`
  
  Gets the weight of this entry.

- `virtual void setInCode (const string inCode)`
  
  Sets input-region code of this entry.

- `virtual void setOutCode (const string outCode)`
  
  Sets output-region code of this entry.

- `virtual void setWeight (const double weight)`
  
  Sets weight of this entry.

- `virtual bool isValid () const`
  
  Checks if this MapperEntry instance contains valid member values.

- `virtual void print () const`
  
  Prints this mapper entry.
4.8.1 Detailed Description

A tuple of in-code, out-code, and weight.

A tuple of in-code, out-code, and weights. Multiple in-code values can be aggregated into one out-code value, or an in-code value could be disaggregated into multiple out-code values, with numerical weights specified in either case.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 MapperEntry::MapperEntry ( const string inCode, const string outCode, const double weight = 1.0 )

MapperEntry Constructor.

Parameters

<table>
<thead>
<tr>
<th>inCode</th>
<th>Input-region code of this entry.</th>
</tr>
</thead>
<tbody>
<tr>
<td>outCode</td>
<td>Output-region code of this entry.</td>
</tr>
<tr>
<td>weight</td>
<td>Weight of this entry. Default to 1.0.</td>
</tr>
</tbody>
</table>

4.8.3 Member Function Documentation

4.8.3.1 string MapperEntry::getInCode ( ) const [virtual]

Gets the input-region code of this entry.

Returns

Input-region code of this entry.

4.8.3.2 string MapperEntry::getOutCode ( ) const [virtual]

Gets the output-region code of this entry.

Returns

Output-region code of this entry.

4.8.3.3 double MapperEntry::getWeight ( ) const [virtual]

Gets the weight of this entry.
4.8 MapperEntry Class Reference

Returns

Weight of this entry.

4.8.3.4 bool MapperEntry::isValid() const [virtual]

Checks if this MapperEntry instance contains valid member values.

Returns

True if this MapperEntry instance contains valid member values, meaning inCode and outCode are not empty strings, and weight is non-negative. False otherwise.

4.8.3.5 void MapperEntry::setInCode(const string inCode) [virtual]

Sets input-region code of this entry.

Parameters


4.8.3.6 void MapperEntry::setOutCode(const string outCode) [virtual]

Sets output-region code of this entry.

Parameters


4.8.3.7 void MapperEntry::setWeight(const double weight) [virtual]

Sets weight of this entry.

Parameters


The documentation for this class was generated from the following files:

- MapperEntry.h
- MapperEntry.cpp

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
4.9 Margins Class Reference

A number-indexed list of margins.

```cpp
#include <Margins.h>
```

Inheritance diagram for Margins:

```
List
/|
V
Margins
```

### Public Member Functions

- **Margins ()**
  
  *Margins default constructor.*

- **Margins (const string)**
  
  *Margins constructor reads an external standardized regions file.*

- **virtual ∼Margins ()**
  
  *Margins default destructor.*

4.9.1 Detailed Description

A number-indexed list of margins.

A numbered list of codes that represent margins.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 **Margins::Margins ( const string filename )**

*Margins constructor reads an external standardized regions file.*

**Parameters**

| `filename` | path to a regions file. |

The documentation for this class was generated from the following files:

- Margins.h
- Margins.cpp
4.10 OscefException Class Reference

Throw exceptions for all core classes in the OSCEF framework.
#include <OscefException.h>
Inherits std::exception.

Public Member Functions

• OscefException ()
  OscefException default constructor.
• OscefException (const string, const string=string(), const size_t=0)
  OscefException constructor.
• virtual ~OscefException () throw ()
  OscefException default destructor.
• virtual void display (ostream &) const
  Displays error message.
• virtual string getErrorMessage () const
  Returns an OSCEF error message.
• virtual string getFilename () const
  Gets the name of the source file where an error occurs.
• virtual size_t getLineNo () const
  Gets the line number of the source file where an error occurs.
• virtual string getMessage () const
  Gets the user-defined error message.
• virtual string getOscefError () const
  Gets OSCEF_EXCEPTION.
• virtual const char ∗ what () const throw ()
  Overrides what() in std::exception.

Protected Member Functions

• virtual void setFilename (const string file)
  Sets the name of the source file where an error occurs.
• virtual void setLineNum (const size_t lineNum)
  Sets the line number of the source file where an error occurs.
• virtual void setMessage (const string msg)
  Sets an user-defined error message.
4.10.1 Detailed Description

Throw exceptions for all core classes in the OSCEF framework.
Throw exceptions for all core classes in the OSCEF framework, with optional error messages, file names, and/or line numbers.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 OscefException::OscefException ( const string message, const string filename = string(), const size_t lineNum = 0 )

OscefException constructor.

Parameters

| message | An user-defined error message. |
| filename | Name of the source file where an error occurs. |
| lineNum | Positive integer. Line number in the source file where an error occurs. |

Example usage:

```
throw OscefException("OSCEF is having trouble opening the file SAMS_USA_2004.csv.",__FILE__,__LINE__); 
```

4.10.3 Member Function Documentation

4.10.3.1 void OscefException::display ( ostream & out ) const [virtual]

Displays error message.

Parameters

| out | * Example usage:

```
try{
  
  } catch (const OscefException& e) {
    e.display(cerr);
  }
```

See also

ostream
4.10.3.2 `string OscefException::getFilename ( ) const [virtual]`

Gets the name of the source file where an error occurs.

Returns

Name of the source file where an error occurs.

4.10.3.3 `size_t OscefException::getLineNo ( ) const [virtual]`

Gets the line number of the source file where an error occurs.

Returns

Line number of the source file where an error occurs.

4.10.3.4 `string OscefException::getMessage ( ) const [virtual]`

Gets the user-defined error message.

Returns

User-defined error message.

4.10.3.5 `string OscefException::getOscefError ( ) const [virtual]`

Gets OSCEF_EXCEPTION.

See also

OSCEF_EXCEPTION

4.10.3.6 `void OscefException::setFilename ( const string filename ) [protected, virtual]`

Sets the name of the source file where an error occurs.

Parameters

| filename | Name of the source file where an error occurs. |

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
### 4.10.3.7 void OscefException::setLineNum ( const size_t lineNum ) [protected, virtual]

Sets the line number of the source file where an error occurs.

**Parameters**

| lineNum | Positive integer. Line number of the source file where an error occurs. Use 0 to signify an invalid line number. |

### 4.10.3.8 void OscefException::setMessage ( const string message ) [protected, virtual]

Sets an user-defined error message.

**Parameters**

| message | Error message. |

### 4.10.3.9 const char * OscefException::what ( ) const throw () [virtual]

Overrides what() in std::exception.

**Note**

This method is called every time an OscefException is thrown.

The documentation for this class was generated from the following files:

- OscefException.h
- OscefException.cpp

## 4.11 PdnFcn Class Reference

Class for reading an input file of standardized production function.

```
#include <PdnFcn.h>
```

Inheritance diagram for PdnFcn:
Public Member Functions

- **PdnFcn ()**
  
  *PdnFcn* default constructor.

- **PdnFcn (const string &)**
  
  *PdnFcn* constructor that reads an external standardized production function.

- **PdnFcn (const string &, const string &)**
  
  *PdnFcn* constructor that reads an external standardized production function.

- **PdnFcn (const CesNode &)**
  
  *PdnFcn* constructor that builds itself on a regional node.

- **PdnFcn (PdnFcn &)**
  
  *PdnFcn* constructor that builds itself on a regional node.

- **virtual ~PdnFcn ()**
  
  *PdnFcn* default destructor.

- **virtual void read (const string)**
  
  Reads an external standardized production function.

- **virtual void extractCesNode (string)**
  
  Extract a CES node from a string in standardized format.

- **virtual void extractCesNode (string, CesNode &)**
  
  Extract a CES node from a string in standardized format.

- **virtual void write (const string)**
  
  Writes an external standardized production function file.

- **virtual void extractTaxes (const Sam &, const string=string())**
  
  Extract taxes into a product tree’s inputs (leaves) from a user-input Sam matrix.

- **virtual void extractExpenses (const Sam &, const string=string())**
  
  Extract expenses into a product tree’s inputs (leaves) from a user-input Sam matrix.

- **virtual void estimateAlphas (const Sam &, const string=string())**
  
  Estimates alphas of a product tree from a user-input Sam matrix.

- **virtual double computeRevenue (const CesNode &)**
  
  Compute revenue of an output as sum of its expense and tax.

- **virtual void computeAlphas ()**
  
  Compute share parameter values in a production tree given expenses on inputs.

- **virtual void computeExpense ()**
  
  Compute expense of an output.
Compute expenses of non-input nodes from the expenses of input nodes.

- virtual void `print()`
  
  *Print this production function.*

- virtual string `printCge(CesNode &CES_NODE = CesNode::EMPTY_CESNODE)`
  
  *Prints the CGE problem statement of an output to a string.*

- virtual string `printMcp(string &S, CesNode &CES_NODE = CesNode::EMPTY_CESNODE)`
  
  *Prints the MCP problem statement of an output to a string.*

- virtual string `printMcp(CesNode &CES_NODE = CesNode::EMPTY_CESNODE)`
  
  *Prints the MCP problem statement of an output to a string.*

- virtual string `printObjective(CesNode &)`
  
  *Prints the objective of maximizing profits in a CGE problem in AMPL syntax.*

- virtual string `printFocObjective(const string = string("USA"), const string = string())`
  
  *Prints the objective of maximizing profits in a CGE problem in AMPL syntax.*

- virtual string `printConstraints(CesNode &)`
  
  *Print the CES constraints of an output and the non-negative constraints of its inputs.*

- virtual string `printFocConstraints(CesNode &S, string &S = string())`
  
  *Print the CES constraints of an output and the non-negative constraints of its inputs.*

- virtual string `printRevenue(const CesNode &SECTOR)`
  
  *Print the after-tax revenue of an output (sector).*

- virtual string `printExpense(CesNode &SECTOR, const string = string(), const string = string())`
  
  *Print the after-tax expense of an input.*

- virtual string `printCesConstraints(CesNode &SECTOR, const string = string(), const string = string())`
  
  *Print the Ces constraints of an output.*

- virtual string `printFocCesConstraints(string &S, CesNode &SECTOR, const string = string(), const string = string())`
  
  *Print the Ces constraints of an output.*

- virtual string `printNonNegativeConstraint(CesNode &SECTOR, const string = string())`
  
  *Print the non-negative constraint of an input.*

- virtual string `printFocNonNegativeConstraint(string &S, CesNode &SECTOR, const string = string())`
  
  *Print the non-negative constraint of an input.*

- virtual string `printOutputVars(const List &L)`
  
  *Print the after-tax revenue of an output (sector).*

- string `printOutputVars(const CesNode &SECTOR)`
  
  *Print the after-tax revenue of an output (sector).*
4.11 PdnFcn Class Reference

Print the after-tax revenue of an output (sector).
• virtual string printInputVars (CesNode &)

Print the after-tax expense of an input.
• virtual CesNode & get (const size_t index)
  Gets the child node by its index value.
• virtual CesNode & get (const string, const bool=true)
  Gets a descendant by its code.
• virtual CesNode & get (bool &found, const string, const bool=true)
  Gets a descendant by its code.
• virtual size_t getSize (const bool=true)
  Returns the number of children or grandchildren.
• void setSigma (const double, const bool)
  Sets the elasticity of substitution of this (input) entry. It also updates rho.

Protected Member Functions

• virtual void setTreeRoot (CesNode treeRoot)
  Sets the root node of the tree.
• virtual CesNode & getTreeRoot ()
  Returns the root node of the tree.
• virtual void setInputsExpenses (CesNode &, const Sectors &, const Factors &, const Double_Sparse_Matrix &, const Double_Sparse_Matrix &, const Double_Sparse_Matrix &, const int)
  Set expenses of inputs in child nodes of an output production tree.
• virtual void setInputsTaxes (CesNode &, const Sectors &, const Factors &, const Double_Sparse_Matrix &, const Double_Sparse_Matrix &, const Double_Sparse_Matrix &, const Double_Sparse_Matrix &, const int)
  Set taxes of inputs in child nodes of an output production tree.

4.11.1 Detailed Description

Class for reading an input file of standardized production function.

Class for reading an input file of standardized production function and building a production function tree using the class CesNode.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 PdnFcn::PdnFcn ( const string & filename )

PdnFcn constructor that reads an external standardized production function.
4.11.2.2  PdnFcn::PdnFcn ( const string & filename, const string & samsPath )

PdnFcn constructor that reads an external standardized production function.

Parameters

<table>
<thead>
<tr>
<th>filename</th>
<th>Path to a production-function specification file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>samsPath</td>
<td>Directory path to a Sam or Sams.</td>
</tr>
</tbody>
</table>

4.11.2.3  PdnFcn::PdnFcn ( const CesNode & regionTree )

PdnFcn constructor that builds itself on a regional node.

Parameters

| regionTree | Adds a regional tree to this PdnFcn instance. |

4.11.2.4  PdnFcn::PdnFcn ( PdnFcn & pf )

PdnFcn constructor that builds itself on a regional node.

Parameters

| pf       | Adds a regional tree to this PdnFcn instance. |

4.11.3  Member Function Documentation

4.11.3.1  double PdnFcn::computeRevenue ( const CesNode & output ) [virtual]

Compute revenue of an output as sum of its expense and tax.

Returns

A number that represents revenue of the output.
### 4.11.3.2 `void PdnFcn::estimateAlphas ( const Sam & sam, const string output = string() ) [virtual]`

Estimates alphas of a product tree from a user-input `Sam` matrix.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sam</code></td>
<td>A SAM matrix.</td>
</tr>
<tr>
<td><code>output</code></td>
<td>Label of targeted production function whose alphas are to be estimated. Optional. Default to empty string, in which case the method will attempt to estimate alphas of all leave nodes in all the production trees associated with the region given by the SAM.</td>
</tr>
</tbody>
</table>

### 4.11.3.3 `void PdnFcn::extractCesNode ( string line ) [virtual]`

Extract a CES node from a string in standardized format.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>line</code></td>
<td>String in standardized format of a CES node.</td>
</tr>
</tbody>
</table>

### 4.11.3.4 `void PdnFcn::extractCesNode ( string line, CesNode & node ) [virtual]`

Extract a CES node from a string in standardized format.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>line</code></td>
<td>String in standardized format of a CES node.</td>
</tr>
<tr>
<td><code>node</code></td>
<td>A CesNode for storing extracted values from input string.</td>
</tr>
</tbody>
</table>

### 4.11.3.5 `void PdnFcn::extractExpenses ( const Sam & sam, const string output = string() ) [virtual]`

Extract expenses into a product tree's inputs (leaves) from a user-input `Sam` matrix.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sam</code></td>
<td>A SAM matrix.</td>
</tr>
<tr>
<td><code>output</code></td>
<td>Label of targeted production function whose alphas are to be estimated.</td>
</tr>
</tbody>
</table>
4.11.3.6  void PdnFcn::extractTaxes ( const Sam & sam, const string output = string() ) [virtual]

Extract taxes into a product tree's inputs (leaves) from a user-input Sam matrix.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sam</td>
<td>A SAM matrix.</td>
</tr>
<tr>
<td>output</td>
<td>Label of targeted production function whose taxes are to be extracted.</td>
</tr>
</tbody>
</table>

4.11.3.7  CesNode & PdnFcn::get ( const size_t index ) [virtual]

Gets the child node by its index value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Index of the child node. Value of index run from 0 to size()-1.</td>
</tr>
</tbody>
</table>

Returns

The child node.

Reimplemented from CesNode.

4.11.3.8  CesNode & PdnFcn::get ( const string code, const bool childrenOnly = true ) [virtual]

Gets a descendant by its code.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>Code of a descendant to look for in this tree.</td>
</tr>
<tr>
<td>childrenOnly</td>
<td>True to conduct search in children only. False to conduct recursive search. Default to true.</td>
</tr>
</tbody>
</table>

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented from CesNode.
4.11.3.9 CesNode & PdnFcn::get ( bool & found, const string code, const bool childrenOnly = true ) [virtual]

Gets a descendant by its code.

Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>found</th>
<th>Code of a descendant to look for in this tree.</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td></td>
<td>True if the node with the input code is found. False otherwise.</td>
</tr>
<tr>
<td>childrenOnly</td>
<td></td>
<td>True to conduct search in children only. False to conduct recursive search. Default to true.</td>
</tr>
</tbody>
</table>

Returns

A descendant node if found. Otherwise, the node itself.

Reimplemented from CesNode.

4.11.3.10 size_t PdnFcn::getSize ( const bool childrenOnly = true ) [virtual]

Returns the number of children or grandchildren.

Parameters

| childrenOnly | If true, returns the number of children. Otherwise, returns the number of grandchildren. Default to true. |

Returns

Number of children or grandchildren.

Reimplemented from CesNode.

4.11.3.11 string PdnFcn::printCesConstraints ( CesNode & node, const string regionLabel = string(), const string outputLabel = string() ) [virtual]

Print the Ces constraints of an output.

Returns

A string expression of the output CES constraints.
4.11.3.12 string PdnFcn::printCge ( CesNode & output = CesNode::EMPTY_CESNODE ) [virtual]

Prints the CGE problem statement of an output to a string.

Parameters

| output | A final output. Optional. Default to print the CGE problem statement of all outputs. |

Returns

string A string expression of the CGE problem statement.

4.11.3.13 string PdnFcn::printConstraints ( CesNode & output ) [virtual]

Print the CES constraints of an output and the non-negative constraints of its inputs.

Returns

A string expression of the constraints.

4.11.3.14 string PdnFcn::printExpense ( CesNode & node, const string regionLabel = string(), const string outputLabel = string() ) [virtual]

Print the after-tax expense of an input.

Parameters

| node | An input node. |
| regionLabel | A string of the region. If empty, minus sign, price and demand variables are not printed. |
| outputLabel | A string of the final output. |
Returns

A string expression of the after-tax expense.

4.11.3.15 string PdnFcn::printFocCesConstraints (string & varlist, CesNode & node, const string regionLabel = string(), const string outputLabel = string()) [virtual]

Print the Ces constraints of an output.

Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>varlist</th>
<th>AMPL declaration of variable(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>node</td>
<td>Product name.</td>
</tr>
<tr>
<td>in</td>
<td>regionLabel</td>
<td>Label for region</td>
</tr>
<tr>
<td>in</td>
<td>outputLabel</td>
<td>Label for output</td>
</tr>
</tbody>
</table>

Returns

A string expression of the output CES constraints.

4.11.3.16 string PdnFcn::printFocConstraints (CesNode & output, string & varlist) [virtual]

Print the CES constraints of an output and the non-negative constraints of its inputs.

Parameters

<table>
<thead>
<tr>
<th>in</th>
<th>output</th>
<th>Product name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>varlist</td>
<td>AMPL declaration of variable(s).</td>
</tr>
</tbody>
</table>

Returns

A string expression of the constraints.

4.11.3.17 string PdnFcn::printFocNonNegativeConstraint (string & varlist, CesNode & node, const string regionLabel = string(), const string outputLabel = string()) [virtual]

Print the non-negative constraint of an input.
Parameters

| in | node | Product name. |
| in | regionLabel | Label for region |
| in | outputLabel | Label for output |

Returns

A string expression of the non-negative constraint of an input.

4.11.3.18 string PdnFcn::printFocObjective ( const string regionLabel = string("USA"), const string sectorLabel = string() ) [virtual]

Prints the objective of maximizing profits in a CGE problem in AMPL syntax.

Parameters

| regionLabel | String of Region Code. Default to "USA". |
| sectorLabel | String of Sector Code. |

Returns

string A string expression of the objective of maximizing profits in a CGE problem.

4.11.3.19 string PdnFcn::printFocObjective ( CesNode & output, string & varlist ) [virtual]

Prints the objective of maximizing profits in a CGE problem in AMPL syntax.

Parameters

| in | output | Product name. |
| out | varlist | AMPL declaration of variable(s). |

Returns

string A string expression of the objective of maximizing profits in a CGE problem.

4.11.3.20 string PdnFcn::printInputVars ( CesNode & node ) [virtual]

Print the after-tax expense of an input.
Returns

A string expression of the after-tax expense.

4.11.3.21 string PdnFcn::printMcp ( string & varlists, CesNode & output = CesNode::EMPTY_CESNODE ) [virtual]

Prints the MCP problem statement of an output to a string.

Parameters

<table>
<thead>
<tr>
<th>in</th>
<th>output</th>
<th>A final output. Optional. Default to print the MCP problem statement of all outputs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>varlists</td>
<td>AMPL variable declaration. Optional. Default to empty string.</td>
</tr>
</tbody>
</table>

Returns

string A string expression of the MCP problem statement.

4.11.3.22 string PdnFcn::printMcp ( CesNode & output = CesNode::EMPTY_CESNODE ) [virtual]

Prints the MCP problem statement of an output to a string.

Parameters

| in     | output | A final output. Optional. Default to print the MCP problem statement of all outputs. |

Returns

string A string expression of the MCP problem statement.

4.11.3.23 string PdnFcn::printNonNegativeConstraint ( CesNode & node, const string regionLabel = string(), const string outputLabel = string() ) [virtual]

Print the non-negative constraint of an input.

Returns

A string expression of the non-negative constraint of an input.
4.11.3.24 string PdnFcn::printObjective ( CesNode & output ) [virtual]

Prints the objective of maximizing profits in a CGE problem in AMPL syntax.

Returns

string A string expression of the objective of maximizing profits in a CGE problem.

4.11.3.25 string PdnFcn::printOutputVars ( const List & list ) [virtual]

Prints the after-tax revenue of an output (sector).

Returns

A string expression of the after-tax revenue.

4.11.3.26 string PdnFcn::printOutputVars ( const CesNode & output )

Prints the after-tax revenue of an output (sector).

Returns

A string expression of the after-tax revenue.

4.11.3.27 string PdnFcn::printRevenue ( const CesNode & output, const string regionLabel = string() ) [virtual]

Prints the after-tax revenue of an output (sector).

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>A final output</td>
</tr>
<tr>
<td>regionLabel</td>
<td>A string of the region. If empty, no price or supply variables printed.</td>
</tr>
</tbody>
</table>

Returns

A string expression of the after-tax revenue.

4.11.3.28 void PdnFcn::read ( const string filename ) [virtual]

Reads an external standardized production function.
4.11 PdnFcn Class Reference

Parameters

| filename | Path to a production-function specification file. Note that it is case sensitive. Preconditions: comma-separated file with first line being and first column being headers, followed by lines and columns of non-negative values, and last row and last column holding the total sum of the column values and row values respectively. |

4.11.3.29 void PdnFcn::setInputsExpenses ( CesNode & outputTree, const Sectors & sectors, const Factors & factors, const Double_Sparse_Matrix & T43, const Double_Sparse_Matrix & T23, const Double_Sparse_Matrix & T13, const int J )
[protected, virtual]

Set expenses of inputs in child nodes of an output production tree.

Parameters

| outputTree | Root node of output production tree. |
| sectors | List of sectors. |
| factors | List of factors. |
| T43 | (4,3)-submatrix of SAMs. |
| T23 | (2,3)-submatrix of SAMs. |
| T13 | (1,3)-submatrix of SAMs. |
| J | Block-column index of the above submatrices. |

4.11.3.30 void PdnFcn::setInputsTaxes ( CesNode & outputTree, const Sectors & sectors, const Factors & factors, const Double_Sparse_Matrix & T83, const Double_Sparse_Matrix & T73, const Double_Sparse_Matrix & T93, const int J )
[protected, virtual]

Set taxes of inputs in child nodes of an output production tree.

Parameters

| outputTree | Root node of output production tree. |
| sectors | List of sectors used in the production. |
| factors | List of factors used in the production. |
| T83 | (8,3)-submatrix of SAMs. |
| T73 | (7,3)-submatrix of SAMs. |
| T93 | (9,3)-submatrix of SAMs. |
| J | Block-column index of the above submatrices. |
4.11.3.31 void PdnFcn::setSigma ( const double sigma, const bool childrenOnly )
   [virtual]

Sets the elasticity of substitution of this (input) entry. It also updates rho.

Parameters

<table>
<thead>
<tr>
<th>sigma</th>
<th>Elasticity of substitution of this (input) entry. Non-negative. To set sigma to positive infinity, use util::INF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>childrenOnly</td>
<td>If true, returns the number of children. Otherwise, returns the number of grandchildren. Default to true.</td>
</tr>
</tbody>
</table>

Reimplemented from CesNode.

4.11.3.32 void PdnFcn::setTreeRoot ( CesNode treeRoot )
   [protected, virtual]

Sets the root node of the tree.

Parameters

| treeRoot | A CesNode instance to become the root of this tree. |

4.11.3.33 void PdnFcn::write ( const string filename )
   [virtual]

Writes an external standardized production function file.

Parameters

| filename | path to the production function file. If the file already existed before, its previous content is deleted and replaced by the new one. |

The documentation for this class was generated from the following files:

- PdnFcn.h
- PdnFcn.cpp

4.12 Regions Class Reference

A number-indexed list of regions.

#include <Regions.h>

Inheritance diagram for Regions:
Public Member Functions

-  **Regions ()**
  
  *Regions* default constructor.

-  **Regions (const string)**
  
  *Regions* constructor reads an external standardized regions file.

-  **virtual ~Regions ()**
  
  *Regions* default destructor.

### 4.12.1 Detailed Description

A number-indexed list of regions.

A numbered list of codes that represent regions.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 Regions::Regions ( const string *filename* )

*Regions* constructor reads an external standardized regions file.

**Parameters**

<table>
<thead>
<tr>
<th><em>filename</em></th>
<th>path to a regions file.</th>
</tr>
</thead>
</table>

The documentation for this class was generated from the following files:

-  Regions.h
-  Regions.cpp

### 4.13 Sam Class Reference

Social accounting matrix in standardized dense or sparse formats.

`#include <Sam.h>`
Public Member Functions

- **Sam** (const int=0)
  
  *Sam default constructor.*

- **Sam** (const string &, const bool=true, const bool=false, const double=7.1)
  
  *Sam constructor reads an external standardized SAM file.*

- **Sam** (const SamDim &, const bool=true)
  
  *Sam constructor creating a zero Sam from a SamDim instance.*

- **virtual ~Sam ()**
  
  *Sam default destructor.*

- **virtual void write** (const string, const bool=false) const
  
  *Writes an external standardized (comma separated) SAM file.*

- **virtual void writeLists** (const string)
  
  *Writes lists of (commas separated) sectors, factors, regions, and margins to external files. Default to false.*

- **virtual string getCode () const**
  
  *Sets the code of a SAM matrix.*

- **virtual size_t getSize () const**
  
  *Gets the size of a SAM matrix.*

- **virtual int getYear () const**
  
  *Gets the year of a SAM matrix.*

- **virtual string getHeader (const size_t) const**
  
  *Gets the header for a given column or row index of a SAM S.*

- **virtual double getValue (const size_t, const size_t) const**
  
  *Gets an element of S, a SAM matrix entry.*

- **virtual void resize** (const size_t, bool=true)
  
  *Resize a SAM matrix to new dimensions.*

- **virtual void setCode** (const string)
  
  *Sets the code of a SAM matrix.*

- **virtual void setYear (int)**
  
  *Sets the code of a SAM matrix.*

- **virtual void setHeader (const size_t, const string)**
  
  *Sets the header for a given column or row of a SAM S.*

- **virtual void setValue (const size_t, const size_t, const double)**
  
  *Sets an element of S, a SAM matrix entry.*

- **virtual void generateHeaders ()**
  
  *Generates headers in standardized format for this Sam.*

- **virtual double getMatSum () const**
  
  *Gets the total of a SAM matrix.*

- **virtual double getColSum (const size_t) const**
4.13 Sam Class Reference

Gets a column sum of the SAM matrix.

- virtual double getRowSum (const size_t) const
  Gets a row sum of a SAM matrix.

- virtual int getNnz () const
  Returns the number of non-zeros in a Sam matrix.

- virtual double getNnzPercent () const
  Returns a number between 0 and 100 inclusive that represents the percentage of non-zeros in a Sam matrix.

- virtual bool isDense () const
  Detects if Sam uses dense or sparse data structure.

- virtual Sam operator+ (const Sam &) const
  Overload operator + for adding two SAMs of the same size.

- virtual Sam operator+ (const double) const
  Overload operator + for multiplying a SAM matrix with a real scalar.

- virtual bool operator==(const Sam &) const
  Overload operator == for comparing two SAMs.

- virtual bool operator!=(const Sam &) const
  Overload operator != for comparing two SAMs.

- virtual void multiply (const size_t, const double, const bool=true)
  Multiply a column or a row in a SAM matrix with a real scalar.

- virtual void computeColSum (const size_t)
  Recompute the sum of a column in the SAM matrix.

- virtual void computeRowSum (const size_t)
  Recompute the sum of a row in the SAM matrix.

- virtual void computeColSums ()
  Recompute the column sums in the SAM matrix.

- virtual void computeRowSums ()
  Recompute the row sums in the SAM matrix.

- virtual void computeMatSum ()
  Recompute the sum of the SAM matrix.

- virtual void computeSums ()
  Recompute the row, column, and matrix sums of the SAM matrix.

- virtual void computeNnz ()
  Count the number of non-zeros and compute the percentage of non-zeros in a Sam matrix.

- virtual void setRowSum (const size_t, const double)
  Sets i-th row sum of S to an input value.

- virtual void setColSum (const size_t, const double)
  Sets j-th row sum of S to an input value.

- virtual void regionAggregate (const string)
Aggregates some region columns according to the input-region mapper’s path.

- virtual void regionAggregate (Mapper=Mapper())
  Aggregates some region columns according to the input region mapper.

- virtual void regionDisaggregate (const string)
  Disaggregate some region columns according to the input region mapper.

- virtual void regionDisaggregate (const Mapper &)=Mapper())
  Disaggregate some region columns according to the input region mapper.

- virtual void sectorAggregate (const string, const bool=true)
  Aggregates a Sam matrix according to the input sector mapper.

- virtual void sectorAggregate (const Mapper &)=Mapper(), const bool=true)
  Aggregates a Sam matrix according to the input sector mapper.

- virtual void marginAggregate (const string, const bool=true)
  Aggregates some columns of marginal commodities according to the input mapper’s path.

- virtual void marginAggregate (const Mapper &)=Mapper(), const bool=true)
  Aggregates some columns of marginal commodities according to the input mapper.

- virtual void factorAggregate (const string)
  Aggregates a Sam matrix according to the input factor mapper.

- virtual void factorAggregate (const Mapper &)=Mapper())
  Aggregates a Sam matrix according to the input factor mapper.

- virtual void sectorDisaggregate (const string)
  Disaggregates a Sam matrix according to the input sector mapper.

- virtual void sectorDisaggregate (const Mapper &)=Mapper())
  Disaggregates a Sam matrix according to the input sector mapper.

- virtual void sectorPermute (const Sectors &)
  Symmetrically permute rows and columns of sectors in this Sam.

- virtual void marginPermute (const Margins &)
  Symmetrically permute rows and columns of margins in this Sam.

- virtual void regionPermute (const Regions &)
  Symmetrically permute rows and columns of regions in this Sam.

- virtual void factorPermute (const Factors &)
  Symmetrically permute rows and columns of factors in this Sam.

- virtual bool isBalanced (const double=0) const
  Test if a Sam matrix is balanced.

- virtual Sam flip () const
  Update a Sam matrix to non-negative.

- virtual Sam computeColCoefMat () const
  Update the column-coefficient matrix of a Sam instance.

- virtual Sam computeRowCoefMat () const
  Update the row-coefficient matrix of a Sam instance.
• virtual bool isColCoeffMat () const
  Check if this Sam is a column-coefficient matrix.
• virtual bool isNonNegative () const
  Check if this Sam is a non-negative matrix.
• virtual Sam balanceByRas () const
  Balance a Sam matrix by iterative row and column scalings.
• virtual Sam balanceByRas (const Double_Vector &) const
  Balance a Sam matrix by iterative row and column scalings.
• virtual Sam balanceByCem () const
  Balance a Sam matrix by cross-entropy method.
• virtual Sam balanceByCem (const Double_Vector &) const
  Balance a Sam matrix by cross-entropy method.
• virtual bool isSizeValid () const
  Checks if size of SAM is valid.
• virtual bool isCodeValid () const
  Checks if code of SAM is valid.
• virtual bool isYearValid () const
  Checks if year of SAM is valid.
• virtual bool isHeadersValid ()
  Checks if headers of SAM is valid.
• virtual bool isMatValid () const
  Checks if the size of S is same as the size of SAM.
• virtual bool isNnzBlocksValid () const
  Check if non-zero blocks is of the OSCEF-defined pattern.
• virtual bool isMatSumsValid (const double=util::EPS, const double=0) const
  Checks if each row sum equals column sum, sum of row sums equals matrix sum, and all sums non-negative.
• virtual bool isValid (const double=util::EPS, const double=0)
  Checks if a SAM object has valid values.
• virtual bool isSparse () const
  Checks if sparse matrix is used.
• virtual void toSparse ()
  Use sparse data structure for storing a SAM matrix.
• virtual void toFull ()
  Use full data structure for storing a SAM matrix.
• virtual void print () const
  Prints the Sam matrix values to standard output.
• virtual void print (const int, const int) const
  Prints a submatrix T(I,J) in this Sam.
virtual Double_Matrix getSubmatrix (const int, const int, const int=0, const int=0) const

Returns a dense submatrix $T(I,J)$ in this Sam.

virtual Double_Sparse_Matrix getSparseSubmatrix (const int, const int, const int=0, const int=0) const

Returns a sparse submatrix $T(I,J)$ in this Sam.

virtual Double_Sparse_Matrix getSparseSubmatrix (const range, const range) const

Returns a sparse submatrix $T(r1,r2)$ in this Sam.

virtual void setSparseSubmatrix (const int, const int, const Double_Sparse_Matrix &)

Sets a sparse submatrix $T$ to this Sam’s block $(I,J)$.

virtual void setSparseSubmatrix (const range r1, const range r2, const Double_Sparse_Matrix &T)

Sets a sparse submatrix $T$ to this Sam’s block $(r1,r2)$, $r1$ and $r2$ are range.

virtual void setSparseSubmatrix (const int, const int, const int, const int, const Double_Sparse_Matrix &)

Sets a sparse submatrix $T$ to this Sam’s block $(I,J)$.

virtual const Double_Matrix & getMatrix () const

Gets a SAM matrix.

virtual const Double_Sparse_Matrix & getSparseMatrix () const

Gets a SAM matrix.

virtual void setSamDim (const SamDim &)

Sets the defining sub-block dimensions of this SAM matrix.

virtual const SamDim & getSamDim () const

Gets a SamDim instance associated with this Sam matrix.

virtual void extractSFRM ()

Extract sectors, regions, factors, and margins from the headers of this Sam.

virtual const Sectors & getSectors () const

virtual const Factors & getFactors () const

virtual const Regions & getRegions () const

virtual const Margins & getMargins () const

virtual void setRegions (const Regions &)

Set regions in Sam.

virtual void setSectors (const Sectors &)

Set sectors in Sam.

virtual void setFactors (const Factors &)

Set factors in Sam.

virtual void setMargins (const Margins &)

Set margin commodities in Sam.

virtual void zero ()
Set all entries in Sam to zero.

- virtual bool isTaxBlock (const int, const int) const
  Check if a submatrix T(I,J) in this Sam is related to taxes.

- virtual Sam toVer8 ()
  Return a SAM before GTAP version 8 to version 8 by deleting the (n-4)th row and column.

- virtual Regions extractRegions (const int=SamDim::regionIndices[0])
  Extract regions from the headers of this Sam.

- virtual Sectors extractSectors (const int=SamDim::sectorIndices[0])
  Extract sectors from the headers of this Sam.

- virtual Factors extractFactors (const int=SamDim::factorIndices[0])
  Extract factors from the headers of this Sam.

- virtual Margins extractMargins (const int=SamDim::marginIndices[0])
  Extract margins from the headers of this Sam.

- virtual void permuteBlock10toRM ()
  Symmetrically permute 10th block row and column of this Sam.

- virtual void permuteBlock10toMR ()
  Symmetrically permute 10th block row and column of this Sam.

- virtual const Double_Sparse_Matrix & getQ_RM () const
  Sets Q_RM used in permuteBlocktoRM().

- virtual void setQ_RM (const Double_Sparse_Matrix &)

- virtual void computeQ_RM ()

Static Public Member Functions

- static int getRasMaxIter ()
  Returns the maximum number of iterations used in balancing by RAS method.

- static double getRasTol ()
  Returns the tolerance used in balancing by RAS method.

- static void setRasMaxIter (const int rasMaxIter)
  Sets the maximum number of iterations used in balancing by RAS method.

- static void setRasTol (const double rasTol)
  Sets the tolerance used in balancing by RAS method.

- static void resetRasTol ()
  Resets the tolerance used in balancing by RAS method to system default.

- static void resetRasMaxIter ()
  Resets the maximum number of iterations used in balancing by RAS method to system default.

- static bool getAutoSparsify ()
Gets the value of whether the software automatically uses sparse data structure for a sparse Sam matrix.

- static void setAutoSparsify (const bool)
  Sets the value of whether the software automatically uses sparse data structure for a sparse Sam matrix.

- static Sam simulateRandomSam (int, bool=false)
  Simulate a full or sparse, non-negative and unbalanced Sam for testing.

Protected Member Functions

- virtual Sam transformSectors (const Sectors &, Double_Sparse_Matrix &)
  Transform this Sam’s sectoral rows and blocks according to a new list of sectors.

- virtual Sam transformFactors (const Factors &, Double_Sparse_Matrix &)
  Transform this Sam’s factorial rows and blocks according to a new list of factors.

- virtual Sam transformRegions (const Regions &, Double_Sparse_Matrix &)
  Transform this Sam’s regional rows and blocks according to a new list of regions.

- virtual Sam transformMargins (const Margins &, Double_Sparse_Matrix &)
  Transform this Sam’s marginal rows and blocks according to a new list of margins.

- virtual bool isRowSumsValid () const
  Checks if the size of row_sum is same as the size of SAM.

- virtual bool isColSumsValid () const
  Checks if the size of row_sum is same as the size of SAM.

- virtual void setMatrix (const Double_Matrix &)
  Sets a SAM matrix.

- virtual void setSparseMatrix (const Double_Sparse_Matrix &)
  Sets a SAM matrix.

4.13.1 Detailed Description

Social accounting matrix in standardized dense or sparse formats.

Social accounting matrix in standardized dense or sparse formats with properties including size, region, year, row sums, column sums, matrix sums, number and percentage of non-zeros.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Sam::Sam ( const int size = 0 )

Sam default constructor.
4.13 Sam Class Reference

Parameters

| size     | Size of this Sam matrix. Default to 0. |

Example usage:

```c
Sam sam;   // a Sam instance of size 0
Sam sam(10); // a Sam instance of size 10
```

4.13.2.2 Sam::Sam (const string & filename, const bool sparse = true, const bool sparseFormat = false, const double version = 7.1 )

Sam constructor reads an external standardized SAM file.

Parameters

<table>
<thead>
<tr>
<th>filename</th>
<th>Path to a SAMS file. Note that it is case sensitive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sparse</td>
<td>Flag to use sparse data structure. Defaults to true.</td>
</tr>
<tr>
<td>sparseFormat</td>
<td>Flag to indicate if the external SAM file is in sparse format. Defaults to false.</td>
</tr>
<tr>
<td>version</td>
<td>GTAP version.</td>
</tr>
</tbody>
</table>

Example usage:

```c
Sam sam_usa("../data/toy3/SAM_USA_2004.csv");
```

4.13.2.3 Sam::Sam (const SamDim & sd, const bool sparse = true )

Sam constructor creating a zero Sam from a SamDim instance.

Parameters

<table>
<thead>
<tr>
<th>sd</th>
<th>A SamDim instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sparse</td>
<td>Flag to use sparse data structure. Defaults to true.</td>
</tr>
</tbody>
</table>

Example usage:

```c
Sam sam_usa(SamDim(4, 3, 2, 1));
```

4.13.3 Member Function Documentation
4.13.3.1  Sam Sam::balanceByCem ( ) const  [virtual]

Balance a Sam matrix by cross-entropy method.

Returns

A balanced Sam matrix, or an empty sam matrix of size 0 if balancing operation
could not be performed due to for example, the Sam matrix is empty, or if the size
of new_sums is not equal to the size of the Sam matrix.

4.13.3.2  Sam Sam::balanceByCem ( const Double Vector & new_sums ) const
[virtual]

Balance a Sam matrix by cross-entropy method.

Parameters

<table>
<thead>
<tr>
<th>new_sums</th>
</tr>
</thead>
</table>
| Optional. Vector of new row and column sums. Default to a vector of
| size 0, in which case the algorithm will work to find a new balanced -
| Sam matrix with row and column sums equal to the average of row and
| column sums of the Sam matrix. Otherwise, the algorithm will work to
| find a new balanced Sam matrix with row and column sums equal to
| new_sums. |

Returns

A balanced Sam matrix, or an empty sam matrix of size 0 if balancing operation
could not be performed due to for example, the Sam matrix is empty, or if the size
of new_sums is not equal to the size of the Sam matrix.

4.13.3.3  Sam Sam::balanceByRas ( ) const  [virtual]

Balance a Sam matrix by iterative row and column scalings.

Returns

A balanced Sam matrix, or this Sam matrix if balancing operation could not be
performed due to for example, the Sam matrix is empty, or if the size of new_sums
is not equal to the size of the Sam matrix.

Example usage:

    Sam sam2 = sam.balanceByRas();
    bool balanced = sam2.isBalanced();
4.13.3.4 Sam Sam::balanceByRas ( const Double_Vector & new_sums ) const [virtual]

Balance a Sam matrix by iterative row and column scalings.

Parameters

| new_sums | Optional. Vector of new row and column sums. Default to a vector of size 0, in which case the algorithm will work to find a new balanced Sam matrix with row and column sums equal to the average of row and column sums of the Sam matrix. Otherwise, the algorithm will work to find a new balanced Sam matrix with row and column sums equal to new_sums. |

Returns

A balanced Sam matrix, or this Sam matrix if balancing operation could not be performed due to for example, the Sam matrix is empty, or if the size of new_sums is not equal to the size of the Sam matrix.

Example usage:

```cpp
Sam sam2 = sam.balanceByRas();
bool balanced = sam2.isBalanced();
```

See also

isBalanced()
4.13.3.6 void Sam::computeColSum ( const size_t j ) [virtual]
Recompute the sum of a column in the SAM matrix.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>Positive integer. Column index.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
int j = 1;
sam.computeColSum(j);
double sum = sam.getColSum(j);
```

See also

- computeColSums(), computeRowSum()

4.13.3.7 void Sam::computeColSums ( ) [virtual]
Recompute the column sums in the SAM matrix.

Example usage:

```cpp
int j = 1;
sam.computeColSums();
double sum = sam.getColSum(j);
```

See also

- computeRowSums(), computeColSum(), getColSum()

4.13.3.8 void Sam::computeMatSum ( ) [virtual]
Recompute the sum of the SAM matrix.

Example usage:

```cpp
sam.computeMatSum();
double sum = sam.getMatSum();
```
4.13 Sam Class Reference

See also

gMatSum(), computeSums(), computeColSums(), computeRowSums(), computeColSum(), computeRowSum()

4.13.3.9 void Sam::computeNnz( ) [virtual]

Count the number of non-zeros and compute the percentage of non-zeros in a Sam matrix.

Note

Call getNnz() or getNnzPercent() to obtain the computed results.

Example usage:

Sam sam("../data/toy1/SAM_USA_2004.csv");
sam.computeNnz();

See also

gNnz(), getNnzPercent()

4.13.3.10 void Sam::computeQ_RM( ) [virtual]

Returns

Computes a class member that is a permutation matrix (of type double and format sparse) that can be used for symmetrically permuting rows and columns of block-10 in this Sam so that margins are grouped under each given region.

See also

gQ_RM(), setQ_RM(), permuteBlocktoRM(), permuteBlocktoMR().

4.13.3.11 Sam Sam::computeRowCoefMat( ) const [virtual]

Update the row-coefficient matrix of a Sam instance.
Returns

The row-coefficient matrix of a Sam, which is defined as the Sam matrix with each (non-zero) row divided by its corresponding row sum.

Example usage:

```c++
Sam sam2 = sam.computeRowCoefMat();
```

See also

`computeColCoefMat()`

4.13.3.12  void Sam::computeRowSum ( const size_t i ) [virtual]

Recompute the sum of a row in the SAM matrix.

Parameters

- **i**: Positive integer. Row index.

Example usage:

```c++
int j = 1;
sam.computeRowSum(j);
double sum = sam.getRowSum(j);
```

See also

`computeRowSums(), computeColSum(),getRowSum()`

4.13.3.13  void Sam::computeRowSums ( ) [virtual]

Recompute the row sums in the SAM matrix.

Example usage:

```c++
int j = 1;
sam.computeRowSums();
double sum = sam.getRowSum(j);
```

See also

`computeRowSum(), computeColSum(),getRowSum()`
4.13.3.14  void Sam::computeSums ( )  [virtual]

Recompute the row, column, and matrix sums of the SAM matrix.
Example usage:

    sam.computeSums();
    double sum = sam.getMatSum();

See also

    getMatSum(), computeMatSum(), computeColSums(), computeRowSums(),
    computeColSum(), computeRowSum()

4.13.3.15  Factors Sam::extractFactors ( const int I = SamDim::factorIndices[0] )
            [virtual]

Extract factors from the headers of this Sam.

Parameters

| I | Positive integer between 1 to 19. Column or row index. Optional. -
|   | Default to 4. |

Returns

A list of factors.

4.13.3.16  Margins Sam::extractMargins ( const int I = SamDim::marginIndices[0] )
            [virtual]

Extract margins from the headers of this Sam.

Parameters

| I | Positive integer between 1 to 19. Column or row index. Optional. -
|   | Default to 11. |

Returns

A list of margins.
4.13.3.17 Regions Sam::extractRegions ( const int I = SamDim::regionIndices[0] )  
[virtual]

Extract regions from the headers of this Sam.

Parameters

| I | Positive integer between 1 to 19. Column or row index. Optional. - Default to 5. |

Returns

A list of regions.

4.13.3.18 Sectors Sam::extractSectors ( const int I = SamDim::sectorIndices[0] )  
[virtual]

Extract sectors from the headers of this Sam.

Parameters

| I | Positive integer between 1 to 19. Column or row index. Optional. - Default to 1. |

Returns

A list of sectors.

4.13.3.19 void Sam::factorAggregate ( const string mapper_path )  [virtual]

Aggregates a Sam matrix according to the input factor mapper.

Parameters

| mapper_path | Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance. |

Returns

this SAM, which has the aggregated factors, or becomes of size 0 if there is no SAM factor to be aggregated.

Example usage:
4.13 Sam Class Reference

```cpp
sam.factorAggregate("../data/mapper/factor_merger_3_1.csv");
```

See also

`factorDisaggregate()`

### 4.13.3.20 void Sam::factorAggregate ( const Mapper & mapper = Mapper () ) [virtual]

Aggregates a Sam matrix according to the input factor mapper.

**Parameters**

- `mapper` A mapper instance. Optional. Defaults to an empty string, which returns the same Sam instance.

**Returns**

this Sam, which has the aggregated factors, or becomes of size 0 if there is no Sam factor to be aggregated.

**Example usage:**

```cpp
Mapper mapper("../data/mapper/factor_merger_3_1.csv");
sam.factorAggregate(mapper);
```

See also

`factorDisaggregate()`

### 4.13.3.21 void Sam::factorPermute ( const Factors & fac ) [virtual]

Symmetrically permute rows and columns of factors in this Sam.

**Parameters**

- `fac` List of factors in new order.

### 4.13.3.22 Sam Sam::flip ( ) const [virtual]

Update a Sam matrix to non-negative.
Returns

A non-negative \texttt{Sam}. If input \(S(i,j)\) is negative, then the output \(R(j,i)\) is set to \(-S(i,j)\) and \(R(i,j)\) to 0. The sum of rows \(i\) and \(j\), columns \(i\) and \(j\), and total matrix sum of \(R\) are updated accordingly.

Example usage:

```cpp
Sam sam2 = sam.flip();
bool is_nonnegative = sam2.isNonNegative(); // true
```

See also

\texttt{isNonNegative()}
4.13 Sam Class Reference

Returns

s Three-letter code of the SAM matrix.

Example usage:

```cpp
string region = sam.getCode(); // "USA"
```

See also

setCode()

4.13.3.26 double Sam::getColSum ( const size_t j ) const [virtual]

Gets a column sum of the SAM matrix.

Parameters

| j | Positive integer. Column index. |

Returns

sum of j-th column of S.

Example usage:

```cpp
double sum = sam.getColSum(1); // first column sum of the SAM
```

See also

computeColSums(), computeColSum(), getRowSum()

4.13.3.27 const Factors & Sam::getFactors ( ) const [virtual]

Returns

A Factors instance that describes this Sam matrix.

4.13.3.28 string Sam::getHeader ( const size_t j ) const [virtual]

Gets the header for a given column or row index of a SAM S.

Parameters

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
| j | Positive integer. Column or row index. |

Returns

Header of the j-th column or row of S.

Example usage:

```
string s = sam.getHeader(1); // header of the first column or first row
```

See also

generateHeaders(), setHeader()

4.13.3.29  const Margins & Sam::getMargins ( ) const  [virtual]

Returns

A Margins instance that describes this Sam matrix.

4.13.3.30  const Double_Matrix & Sam::getMatrix ( ) const  [virtual]

Gets a SAM matrix.

Returns

boost::numeric::ublas::matrix that represents the SAM matrix.

4.13.3.31  double Sam::getMatSum ( ) const  [virtual]

Gets the total of a SAM matrix.

Returns

Total sum of the SAM matrix.

Example usage:

```
double sum = sam.getMatSum();
```

See also

computeMatSum(), getColSum(), getRowSum()
4.13.3.32 int Sam::getNnz ( ) const [virtual]
Returns the number of non-zeros in a Sam matrix.

Note
Call computeNnz() to compute the number of non-zeros.

Example usage:

    size_t nnz = sam.getNnz(); // number of non-zeros in sam

See also
    computeNnz(), getNnzPercent()

4.13.3.33 double Sam::getNnzPercent ( ) const [virtual]
Returns a number between 0 and 100 inclusive that represents the percentage of non-zeros in a Sam matrix.

Note
Call computeNnz() to compute the percentage of non-zeros.

Example usage:

    double nnzp = sam.getNnzPercent(); // percentage of non-zeros in sam

See also
    getNnz(), getNnzPercent()

4.13.3.34 const Double_Sparse_Matrix & Sam::getQ_RM ( ) const [virtual]
Returns
    A permutation matrix (of type double and format sparse) that can be used for symmetrically permuting rows and columns of block-10 in this Sam so that margins are grouped under each given region.

See also
    setQ_RM(), computeQ_RM(), permuteBlocktoRM(), permuteBlocktoMR().
4.13.3.35 int Sam::getRasMaxIter() [static]

Returns the maximum number of iterations used in balancing by RAS method.
Example usage:

    int max_iter = Sam::getRasMaxIter();

See also

    setRasMaxIter(), resetRasMaxIter()

4.13.3.36 double Sam::getRasTol() [static]

Returns the tolerance used in balancing by RAS method.
Example usage:

    double tol = Sam::getRasTol();

See also

    setRasTol(), resetRasTol()

4.13.3.37 const Regions & Sam::getRegions() const [virtual]

Returns

    A Regions instance that describes this Sam matrix.

4.13.3.38 double Sam::getRowSum(const size_t i) const [virtual]

Gets a row sum of a SAM matrix.

Parameters

    i | Positive integer. Row index.

Returns

    sum of i-th row of S.

Example usage:
double sum = getRowSum(1); // first row sum of the SAM

See also
computeRowSums(), computeRowSum(), getColSum()

4.13.39  const SamDim & Sam::getSamDim ( ) const  [virtual]

Gets a SamDim instance associated with this Sam matrix.
Returns
A SamDim instance that defines the submatrix structure of this SAM.

4.13.40  const Sectors & Sam::getSectors ( ) const  [virtual]

Returns
A Sectors instance that describes this Sam matrix.

4.13.41  size_t Sam::getSize ( ) const  [virtual]

Gets the size of a SAM matrix.
Returns
size of a SAM matrix.
Example usage:

    int i = sam.getSize();

See also
resize()

4.13.42  const Double_Sparse_Matrix & Sam::getSparseMatrix ( ) const  [virtual]

Gets a SAM matrix.
Returns
boost::numeric::ublas::compressed_matrix that represents the SAM matrix.
4.13.3.43 Double_Sparse_Matrix Sam::getSparseSubmatrix ( const int \( I \), const int \( J \), const int \( K = 0 \), const int \( L = 0 \) ) const [virtual]

Returns a sparse submatrix \( T(I,J) \) in this \textit{Sam}.

Parameters

\begin{center}
\begin{tabular}{|c|}
\hline
\textit{I}  \\
\textit{J}  \\
\textit{K}  \\
\textit{L}  \\
\hline
\end{tabular}
\end{center}

- \textit{I} Positive integer between 1 to 19.
- \textit{J} Positive integer between 1 to 19.
- \textit{K} Positive integer between 1 to \( M \) if \( I = 10 \), else is (default to) 0.
- \textit{L} Positive integer between 1 to \( M \) if \( J = 10 \), else is (default to) 0.

See also

\textit{SamDim}

4.13.3.44 Double_Sparse_Matrix Sam::getSparseSubmatrix ( const range \( r1 \), const range \( r2 \) ) const [virtual]

Returns a sparse submatrix \( T(r1,r2) \) in this \textit{Sam}.

Parameters

\begin{center}
\begin{tabular}{|c|}
\hline
\textit{r1}  \\
\textit{r2}  \\
\hline
\end{tabular}
\end{center}

- \textit{r1} Range associated with rows.
- \textit{r2} Range associated with columns.

See also

\textit{SamDim}, \textit{ublas::range}

4.13.3.45 Double_Matrix Sam::getSubmatrix ( const int \( I \), const int \( J \), const int \( K = 0 \), const int \( L = 0 \) ) const [virtual]

Returns a dense submatrix \( T(I,J) \) in this \textit{Sam}.

Parameters

\begin{center}
\begin{tabular}{|c|}
\hline
\textit{I}  \\
\textit{J}  \\
\textit{K}  \\
\textit{L}  \\
\hline
\end{tabular}
\end{center}

- \textit{I} Positive integer between 1 to 19.
- \textit{J} Positive integer between 1 to 19.
- \textit{K} Positive integer between 1 to \( M \) if \( I = 10 \), else is (default to) 0.
- \textit{L} Positive integer between 1 to \( M \) if \( J = 10 \), else is (default to) 0.
4.13.3.46 double Sam::getValue ( const size_t i, const size_t j ) const  [virtual]

Gets an element of S, a SAM matrix entry.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Positive integer. Row index of the element.</td>
</tr>
<tr>
<td>j</td>
<td>Positive integer. Column index of the element.</td>
</tr>
</tbody>
</table>

Returns

S(i,j).

Example usage:

   double v = sam.getValue(1,10); // value of S(1,10)

See also

setValue()

4.13.3.47 int Sam::getYear ( ) const  [virtual]

Gets the year of a SAM matrix.

Returns

Year of the SAM matrix.

Example usage:

   int year = sam.getYear();

See also

setYear()
4.13.3.48 bool Sam::isBalanced ( const double tol = 0 ) const [virtual]

Test if a Sam matrix is balanced.

Parameters

- `tol` Small positive number. Default to machine precision.

Returns

True if each row sum equals its corresponding column sum with a precision up to about the most significant log10(tol) digits.

Note

Consider calling computeSums() before calling this method.

Example usage:

```cpp
sam.computeSums();
bool balanced = sam.isBalanced();

sam.computeSums();
bool balanced = sam.isBalanced(1e-5);
```

See also

- balanceByRas()

4.13.3.49 bool Sam::isCodeValid ( ) const [virtual]

Checks if code of SAM is valid.

Returns

True if code of SAM is not an empty string or one with only white spaces.

Example usage:

```cpp
bool valid = sam.isCodeValid();
```

See also

- isValid(), isYearValid(), isHeadersValid(), isMatValid()
4.13 Sam Class Reference

4.13.3.50 bool Sam::isColCoefMat() const [virtual]

Check if this Sam is a column-coefficient matrix.

Returns

True if Sam is a column-coefficient matrix, which is one with each column sum equal to one.

Example usage:

```cpp
generate.isCCM = sam.isColCoefMat();
```

See also

`computeColCoefMat()`

4.13.3.51 bool Sam::isColSumsValid() const [protected, virtual]

Checks if the size of row_sum is same as the size of SAM.

Returns

True if the size of row_sum is same as the size of SAM.

4.13.3.52 bool Sam::isDense() const [virtual]

Detects if Sam uses dense or sparse data structure.

Returns

True if this Sam instance uses a dense matrix for storage.

Example usage:

```cpp
generate.is dense = sam.isDense();
```

See also

`isSparse()`
4.13.3.53  bool Sam::isHeadersValid ( ) [virtual]

Checks if headers of SAM is valid.

Returns

True if all headers are not empty strings.

4.13.3.54  bool Sam::isMatSumsValid ( const double tol = util::EPS, const double drop = 0 ) const [virtual]

Checks if each row sum equals column sum, sum of row sums equals matrix sum, and all sums non-negative.

Parameters

<table>
<thead>
<tr>
<th>tol</th>
<th>Relative tolerance for testing if two numbers are equal. Optional. - Default to machine precision.</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop</td>
<td>If the magnitude of one or both numbers is less than the drop, we do not hold them to the relative-tolerance test. Optional. Default to 0.</td>
</tr>
</tbody>
</table>

Returns

Returns true or false.

4.13.3.55  bool Sam::isMatValid ( ) const [virtual]

Checks if the size of S is same as the size of SAM.

Returns

True if the size of S is same as the size of SAM.

4.13.3.56  bool Sam::isNnzBlocksValid ( ) const [virtual]

Check if non-zero blocks is of the OSCEF-defined pattern.

Returns

True if pattern is fitting, false otherwise.
4.13.3.57  bool Sam::isNonNegative ( ) const  [virtual]

Check if this Sam is a non-negative matrix.

Returns
True if every entry in this Sam matrix is non-negative.

Example usage:

    bool non_neg = sam.isNonNegative();

See also
computeColCoefMat()

4.13.3.58  bool Sam::isRowSumsValid ( ) const  [protected, virtual]

Checks if the size of row_sum is same as the size of SAM.

Returns
True if the size of row_sum is same as the size of SAM.

4.13.3.59  bool Sam::isSizeValid ( ) const  [virtual]

Checks if size of SAM is valid.

Returns
True if it is positive and contains at least one sector, region, factor, and margin.

Example usage:

    bool valid = sam.isSizeValid();

See also
isCodeValid(), isYearValid(), isHeadersValid(), isMatValid()

4.13.3.60  bool Sam::isSparse ( ) const  [virtual]

Checks if sparse matrix is used.

Returns
True if sparse matrix is used.
4.13.3.61 bool Sam::isTaxBlock ( const int I, const int J ) const [virtual]

Check if a submatrix T(I,J) in this Sam is related to taxes.

Parameters

| I | Positive integer between 1 to 19. |
| J | Positive integer between 1 to 19. |

Returns

Returns true if the submatrix is about taxes, false otherwise.

4.13.3.62 bool Sam::isValid ( const double tol = util::EPS, const double drop = 0 ) [virtual]

Checks if a SAM object has valid values.

Parameters

| tol | Relative tolerance for testing if two numbers are equal in the method isMatSumsValid(). Optional. Default to machine precision. |
| drop | If the magnitude of one or both numbers is less than the drop, in isMatSumsValid(), we do not hold them to the relative-tolerance test. - Optional. Default to 0. |

Returns

True if this SAM object has all values valid.

4.13.3.63 bool Sam::isYearValid ( ) const [virtual]

Checks if year of SAM is valid.

Returns

True if year is positive.

4.13.3.64 void Sam::marginAggregate ( const string mapper_path, const bool ifSectorAgg = true ) [virtual]

Aggregates some columns of marginal commodities according to the input mapper’s path.
4.13 Sam Class Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapper_path</td>
<td>path to a mapper file. Optional. Defaults to an empty string, summing all the sam matrices.</td>
</tr>
<tr>
<td>ifSectorAgg</td>
<td>Set it to false to stop aggregation of margins in blocks related to sectors. Optional. Default to true.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
marginAggregate("./data/mapper/margin_merger_3_1.csv");
```

Note: Margins should be a strict subset of sectors.

4.13.3.65 void Sam::marginAggregate ( const Mapper & mapper = Mapper (), const bool ifSectorAgg = true ) [virtual]

Aggregates some columns of marginal commodities according to the input mapper.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapper</td>
<td>A mapper instance. Optional. Defaults to an empty string, summing all the sam matrices.</td>
</tr>
<tr>
<td>ifSectorAgg</td>
<td>Set it to false to stop aggregation of margins in blocks related to sectors. Optional. Default to true.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
Mapper mapper;
mapper.setMapperEntry(MapperEntry("TRL", "LAS"));
mapper.setMapperEntry(MapperEntry("TRA", "LAS"));
mapper.setMapperEntry(MapperEntry("TRS", "LAS"));
sam.marginAggregate(mapper);
```

Note: Margins should be a strict subset of sectors.

4.13.3.66 void Sam::marginPermute ( const Margins & mar ) [virtual]

Symmetrically permute rows and columns of margins in this Sam.
4.13.3.67 void Sam::multiply( const size_t j, const double c, const bool co = true )
    [virtual]

Multiply a column or a row in a SAM matrix with a real scalar.

Parameters

<table>
<thead>
<tr>
<th>j</th>
<th>Column or row index, between 1 and size of Sam.</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Scalar.</td>
</tr>
<tr>
<td>co</td>
<td>Column-oriented. Default to true. Row-oriented if set to false.</td>
</tr>
</tbody>
</table>

Note

May need to call computeSums() to update column, row, and matrix sums.

Example usage:

```c++
int j = 1;
sam.computeColSum(j);
sam.multiply(j, 0.5); // multiply the first column by 0.5
```

See also

operator*( ), computeColSum()

4.13.3.68 bool Sam::operator!=( const Sam & sam2 ) const  [virtual]

Overload operator != for comparing two SAMs.

sam1 Left-hand-side SAM matrix.

Parameters

| sam2 | Right-hand-side SAM matrix. |

Returns

False if sam1 and sam2 have the same size, code, year, headers, SAM , values, and sums (row sums, column sums, matrix sum).

Example usage:
bool diff = (sam1 != sam2); // true if S1 equals S2

See also
operator==(), operator+, operator+()

4.13.3.69 Sam Sam::operator+ ( const double c ) const [virtual]
Overload operator + for multiplying a SAM matrix with a real scalar.

Parameters

| c  | Scalar. |

Returns
Product of sam and c as a Sam matrix. The headers of the aggregated matrix are the same the headers of the input SAM matrix. The row, column, and matrix sums are computed for the product.

Example usage:

Sam sam2 = sam * 0.5; // S2(i,j) = 0.5 * S(i,j) for all i and j

See also
operator==(), operator+()

4.13.3.70 Sam Sam::operator+ ( const Sam & sam2 ) const [virtual]
Overload operator + for adding two SAMs of the same size.

Parameters

| sam2 | Second SAM matrix. |

Returns
Sum of sam1 and sam2 as a Sam matrix. The headers of the aggregated matrix are the same the headers of the first SAM matrix. The row, column, and matrix sums are computed from the respective sums of the two input matrices.
Example usage:

```c++
sam1 = sam1 + sam2;
```

See also

`operator==(), operator*()`

4.13.3.71 bool Sam::operator== ( const Sam & sam2 ) const [virtual]

Overload operator == for comparing two SAMs.

`sam1` Left-hand-side SAM matrix.

**Parameters**

| `sam2` | Right-hand-side SAM matrix. |

**Returns**

True if `sam1` and `sam2` have the same size, code, year, headers, SAM, values, and sums (row sums, column sums, matrix sum).

Example usage:

```c++
bool equal = (sam1 == sam2); // true if S1 equals S2
```

See also

`operator!=( ), operator* ( ), operator+ ( )`

4.13.3.72 void Sam::permuteBlock10toMR ( ) [virtual]

Symmetrically permute 10th block row and column of this `Sam`.

**Returns**

Block-10 rows/columns will have regions grouped together under each margin.

**Note**

This is not the default order in a GTAP `Sam`.

See also

`: permuteBlock10toRM`
4.13.3.73  void Sam::permuteBlock10toRM( ) [virtual]

Symmetrically permute 10th block row and column of this Sam.

Returns

Block-10 rows/columns will have margins grouped together under each region.

Note

This is not the default order in a GTAP Sam.

See also

: permuteBlock10toMR()

4.13.3.74  void Sam::print ( const int I, const int J ) const [virtual]

Prints a submatrix T(I,J) in this Sam.

Parameters

<table>
<thead>
<tr>
<th>I</th>
<th>Positive integer between 1 to 19.</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>Positive integer between 1 to 19.</td>
</tr>
</tbody>
</table>

See also

SamDim

4.13.3.75  void Sam::regionAggregate ( const string mapper_path ) [virtual]

Aggregates some region columns according to the input-region mapper's path.

Parameters

| mapper_path | path to a mapper file. Optional. Defaults to an empty string, summing all the sam matrices. |

Example usage:

```
sam.regionAggregate("../data/mapper/mapping_merger_3_1a.csv");
```
4.13.3.76  void Sam::regionAggregate ( Mapper mapper = Mapper() ) [virtual]

Aggregates some region columns according to the input region mapper.

Parameters

- **mapper**: A mapper instance. Optional. Defaults to an empty string, summing all the sam matrices.

Returns

- a SAM, which is the sum of all SAMs; or a SAM of size 0 if there is no SAM to be summed.

Example usage:

```cpp
Mapper mapper;
mapper.setMapperEntry(MapperEntry("USA", "USA"));
mapper.setMapperEntry(MapperEntry("CAN", "ROW"));
mapper.setMapperEntry(MapperEntry("AUS", "ROW"));
sam.regionAggregate(mapper);
```

See also

- regionDisaggregate()

4.13.3.77  void Sam::regionDisaggregate ( const string mapper_path ) [virtual]

Disaggregate some region columns according to the input region mapper.

Parameters

- **mapper_path**: path to a mapper file. Optional. Defaults to an empty string, summing all the sam matrices.

Returns

- a SAM, which is the sum of all SAMs; or a SAM of size 0 if there is no SAM to be split.

Example usage:
Sam::regionDisaggregate (const Mapper & mapper = Mapper()) [virtual]

Disaggregate some region columns according to the input region mapper.

Parameters

mapper A mapper instance. Optional. Defaults to an empty string, returning the same sam instance.

Returns

a SAM, which is the sum of all SAMs; or a SAM of size 0 if there is no SAM to be split.

Example usage:

Mapper mapper("../data/mapper/mapping_demerg_1_3b.csv");
sam.regionDisaggregate(mapper);

See also

regionAggregate()
4.13.3.80  void Sam::resetRasMaxIter() [static]

Resets the maximum number of iterations used in balancing by RAS method to system default.
Example usage:

    Sam::resetRasMaxIter();

See also
    getRasMaxIter(), setRasMaxIter()

4.13.3.81  void Sam::resetRasTol() [static]

Resets the tolerance used in balancing by RAS method to system default.
Example usage:

    Sam::resetRasTol();

See also
    getRasTol(), setRasTol()

4.13.3.82  void Sam::resize( const size_t newSize, bool preserve = true ) [virtual]

Resize a SAM matrix to new dimensions.

Parameters

<table>
<thead>
<tr>
<th>newSize</th>
<th>new size of a SAM matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>preserve</td>
<td>An optional, boolean variable. True to preserve data in the original matrix. Default to true. If the new size is bigger, extra values are filled with zeros. If the new size is smaller, last values are lost. If data are not preserved, then nothing has to be assumed regarding the content of the matrix after resizing.</td>
</tr>
</tbody>
</table>

Warning

This method may result in inconsistent row, column, and matrix sums of the Sam. Call the method computeSums() to recompute the sums.

Example usage:

    Sam::resetRasMaxIter();
sam.resize(sam.getSize() * 2); // twice the original size

See also
getSize()

4.13.3.83  void Sam::sectorAggregate ( const string mapper_path, const bool ifMarginAgg = true ) [virtual]

Aggregates a Sam matrix according to the input sector mapper.

Parameters

<table>
<thead>
<tr>
<th>mapper_path</th>
<th>Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifMarginAgg</td>
<td>whether to perform margin aggregation. Default to true.</td>
</tr>
</tbody>
</table>

Returns

this SAM, which has the aggregated sectors, or becomes of size 0 if there is no SAM sector to be aggregated.

Example usage:

sam.sectorAggregate("./data/mapper/sector_merge_3_1.csv");

See also
sectorDisaggregate()

4.13.3.84  void Sam::sectorAggregate ( const Mapper & mapper = Mapper (), const bool ifMarginAgg = true ) [virtual]

Aggregates a Sam matrix according to the input sector mapper.

Parameters

<table>
<thead>
<tr>
<th>mapper</th>
<th>A mapper instance. Optional. Defaults to an empty string, which returns the same sam instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifMarginAgg</td>
<td>whether to perform margin aggregation. Default to true.</td>
</tr>
</tbody>
</table>
Returns

this SAM, which has the aggregated sectors, or becomes of size 0 if there is no SAM sector to be aggregated.

Example usage:

```cpp
Mapper mapper("../data/mapper/sector_merger_3_1.csv");
sam.sectorAggregate(mapper);
```

See also

sectorDisaggregate()

4.13.3.85 void Sam::sectorDisaggregate ( const string mapper_path ) [virtual]

Disaggregates a Sam matrix according to the input sector mapper.

Parameters

| mapper_path | Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance. |

Returns

this SAM, which has the disaggregated sectors, or becomes of size 0 if there is no SAM sector to be disaggregated.

Example usage:

```cpp
sam.sectorDisaggregate("../data/mapper/sector_demmerger_1_3.csv");
```

See also

sectorAggregate()

4.13.3.86 void Sam::sectorDisaggregate ( const Mapper & mapper = Mapper() ) [virtual]

Disaggregates a Sam matrix according to the input sector mapper.

Parameters

| mapper | A mapper instance. Optional. Defaults to an empty string, which returns the same sam instance. |
4.13 Sam Class Reference

Returns

this SAM, which has the disaggregated sectors, or becomes of size 0 if there is no
SAM sector to be disaggregated.

Example usage:

Mapper mapper("../data/mapper/sector_demerger_1_3.csv");
sam.sectorDisaggregate(mapper);

See also

sectorAggregate()

4.13.3.87 void Sam::sectorPermute ( const Sectors & sec ) [virtual]

Symmetrically permute rows and columns of sectors in this Sam.

Parameters

| sec | List of sectors in new order. |

4.13.3.88 void Sam::setAutoSparsify ( const bool autoSparsify ) [static]

Sets the value of whether the software automatically uses sparse data structure for a
sparse Sam matrix.

Parameters

| autoSparsify | Sets it to true if the software automatically uses sparse data structure for a sparse Sam matrix. |

4.13.3.89 void Sam::setCode ( const string s ) [virtual]

Sets the code of a SAM matrix.

Parameters

| s | Three-letter code of the SAM matrix. |

Example usage:

    sam.setCode("USA");
4.13.3.90 void Sam::setColSum ( const size_t j, const double value ) [virtual]

Sets j-th row sum of S to an input value.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>Positive integer. Column index of the element.</td>
</tr>
<tr>
<td>value</td>
<td>Assign value to the j-th column sum of S.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
int i = 10;
setColSum(i, 0); // column-10 has sum 0
```

See also

gColSum(), getRowSum()

4.13.3.91 void Sam::setFactors ( const Factors & f ) [virtual]

Set factors in Sam.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>A list of factors.</td>
</tr>
</tbody>
</table>

4.13.3.92 void Sam::setHeader ( const size_t j, const string s ) [virtual]

Sets the header for a given column or row of a SAM S.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>Positive integer. Column or row index.</td>
</tr>
<tr>
<td>s</td>
<td>Column or row label.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
sam2.setHeader(1, sam1.getHeader(1));
// sam1 and sam2 now have the same first column and row headers
```
See also
   generateHeaders(), getHeader()

4.13.3.93  void Sam::setMargins ( const Margins & m ) [virtual]
Set margin commodities in Sam.
Parameters
   m  A list of margin commodities.

4.13.3.94  void Sam::setMatrix ( const DoubleMatrix & M ) [protected, virtual]
Sets a SAM matrix.
Parameters
   M  boost::numeric::ublas::matrix. Values of a SAM.

4.13.3.95  void Sam::setQ_RM ( const DoubleSparseMatrix & Q ) [virtual]
Sets Q_RM used in permuteBlocktoRM().
Parameters
   Q  boost::numeric::ublas::compressed_matrix. A permutation matrix of size R+M.

4.13.3.96  void Sam::setRasMaxIter ( const int rasMaxIter ) [static]
Sets the maximum number of iterations used in balancing by RAS method.
Parameters
   rasMaxIter  Positive integer.

Example usage:

   Sam::setRasMaxIter(sam.getSize());
4.13.3.97  void Sam::setRasTol ( const double rasTol ) [static]

Sets the tolerance used in balancing by RAS method.

Parameters

| rasTol | Positive small number. |

Example usage:

```c++
Sam::setRasTol(1e-4);
```

See also

getRasMaxIter(), resetRasMaxIter()

4.13.3.98  void Sam::setRegions ( const Regions & r ) [virtual]

Set regions in Sam.

Parameters

| r | A list of regions. |

4.13.3.99  void Sam::setRowSum ( const size_t i, const double value ) [virtual]

Sets i-th row sum of S to an input value.

Parameters

| i | Positive integer. Row index of the element. |
| value | Assign value to the i-th row sum of S. |

Example usage:

```c++
int i = 10;
setRowSum(i, 0);  // row-10 has sum 0
```
See also
getRowSum(), getColSum()

### 4.13.3.100 void Sam::setSamDim ( const SamDim & samDim ) [virtual]
Sets the defining sub-block dimensions of this SAM matrix.

**Parameters**

| samDim | A SamDim instance. |

### 4.13.3.101 void Sam::setSectors ( const Sectors & s ) [virtual]
Set sectors in Sam.

**Parameters**

| s | A list of sectors. |

### 4.13.3.102 void Sam::setSparseMatrix ( const Double_Sparse_Matrix & M )
[protected, virtual]
Sets a SAM matrix.

**Parameters**

| M | boost::numeric::ublas::compressed_matrix. Values of a SAM. |

### 4.13.3.103 void Sam::setSparseSubmatrix ( const int I, const int J, const Double_Sparse_Matrix & T ) [virtual]
Sets a sparse submatrix T to this Sam's block (I,J).

**Parameters**

| I | Positive integer between 1 to 19. |
| J | Positive integer between 1 to 19. |
| T | Double_Sparse_Matrix. |
See also

SamDim

4.13.3.104  void Sam::setSparseSubmatrix ( const range r1, const range r2, const
Double_Sparse_Matrix & T ) [virtual]

Sets a sparse submatrix T to this Sam’s block (r1,r2), r1 and r2 are range.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Range associated with rows.</td>
</tr>
<tr>
<td>r2</td>
<td>Range associated with columns.</td>
</tr>
<tr>
<td>T</td>
<td>Double_Sparse_Matrix.</td>
</tr>
</tbody>
</table>

See also

SamDim, ublas::range

4.13.3.105  void Sam::setSparseSubmatrix ( const int I, const int J, const int K, const int L, const Double_Sparse_Matrix & T ) [virtual]

Sets a sparse submatrix T to this Sam’s block (I,J).

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Positive integer between 1 to L.</td>
</tr>
<tr>
<td>J</td>
<td>Positive integer between 1 to L.</td>
</tr>
<tr>
<td>K</td>
<td>Positive integer between 1 to M.</td>
</tr>
<tr>
<td>L</td>
<td>Positive integer between 1 to M.</td>
</tr>
<tr>
<td>T</td>
<td>Double_Sparse_Matrix.</td>
</tr>
</tbody>
</table>

See also

SamDim::getL, SamDim::getM

4.13.3.106  void Sam::setValue ( const size_t i, const size_t j, const double value )
[virtual]

Sets an element of S, a SAM matrix entry.
### 4.13 Sam Class Reference

#### Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Positive integer. Row index of the element.</td>
</tr>
<tr>
<td>j</td>
<td>Positive integer. Column index of the element.</td>
</tr>
<tr>
<td>value</td>
<td>Assign value to S(i,j).</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
int i = 10, j = 10;
sam2.setValue(i, j, sam1.getValue(i, j) * 10); // S1(10,10) = S2(10,10) * 10
```

See also

- `getValue()`

#### 4.13.3.107 void Sam::setYear ( int year ) [virtual]

Sets the code of a SAM matrix.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Year of the SAM matrix.</td>
</tr>
</tbody>
</table>

Example usage:

```cpp
sam.setYear(2012);
```

See also

- `getYear()`

#### 4.13.3.108 Sam Sam::simulateRandomSam ( int size, bool sparse = false ) [static]

Simulate a full or sparse, non-negative and unbalanced Sam for testing.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>Non-negative integer. Size of the simulated Sam.</td>
</tr>
<tr>
<td>sparse</td>
<td>Set it to true to simulate a sparse Sam matrix, in which case the percentage of non-zeros in the matrix is returned by <code>getNnzPercent()</code>;</td>
</tr>
</tbody>
</table>
4.13.3.109 Sam::toVer8()
[virtual]

Return a SAM before GTAP version 8 to version 8 by deleting the (n-4)th row and column.

Note
The 8.0 version of GTAP aggregator drops the SALTAX row and column.

4.13.3.110 Sam::transformFactors(const Factors& fac, Double_Sparse_Matrix& Q)
[protected, virtual]

Transform this Sam's factorial rows and blocks according to a new list of factors.

Parameters

<table>
<thead>
<tr>
<th>fac</th>
<th>New list of factors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>An orthogonal matrix that corresponds to renaming, permuting, or aggregating factorial rows and columns to the new order.</td>
</tr>
</tbody>
</table>

Returns

A Sam instance with the new factors if transformation is successful. If inputs are erroneous or the operation cannot be completed successfully, this Sam instance is returned.

4.13.3.111 Sam::transformMargins(const Margins& mar, Double_Sparse_Matrix& Q)
[protected, virtual]

Transform this Sam's marginal rows and blocks according to a new list of margins.

Parameters

<table>
<thead>
<tr>
<th>mar</th>
<th>New list of margins.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>An orthogonal matrix that corresponds to renaming, permuting, or aggregating marginal rows and columns to the new order.</td>
</tr>
</tbody>
</table>

Returns

A Sam instance with the new margins if transformation is successful. If inputs are erroneous or the operation cannot be completed successfully, this Sam instance is returned.
4.13.3.112  Sam::transformRegions (const Regions &reg, Double_Sparse_Matrix &Q) [protected, virtual]

Transform this Sam's regional rows and blocks according to a new list of regions.

Parameters

<table>
<thead>
<tr>
<th>reg</th>
<th>New list of regions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>An orthogonal matrix that corresponds to renaming, permuting, or aggregating regional rows and columns to the new order.</td>
</tr>
</tbody>
</table>

Returns

A Sam instance with the new regions if transformation is successful. If inputs are erroneous or the operation cannot be completed successfully, this Sam instance is returned.

4.13.3.113  Sam::transformSectors (const Sectors &sec, Double_Sparse_Matrix &Q) [protected, virtual]

Transform this Sam's sectoral rows and blocks according to a new list of sectors.

Parameters

<table>
<thead>
<tr>
<th>sec</th>
<th>New list of sectors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>An orthogonal matrix that corresponds to renaming, permuting, or aggregating sectoral rows and columns to the new order.</td>
</tr>
</tbody>
</table>

Returns

A Sam instance with the new sectors if transformation is successful. If inputs are erroneous or the operation cannot be completed successfully, this Sam instance is returned.

4.13.3.114  void write (const string filename, const bool sparseFormat = false) const [virtual]

Writes an external standardized (comma separated) SAM file.

Parameters

<table>
<thead>
<tr>
<th>filename</th>
<th>path to the SAMS file. If the file already existed before, its previous content is deleted and replaced by the new one.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sparseFormat</td>
<td>Flag to indicate that the output file will be written in sparse-matrix format. Default to false.</td>
</tr>
</tbody>
</table>

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
Example usage:

```cpp
Sam sam2 = sam.balanceByRas();
sam2.write("../data/toy6/out/SAM_USA_2004-pos-sums.csv");
```

4.13.3.115  void Sam::writeLists ( const string `pathname` ) [virtual]

Writes lists of (commas separated) sectors, factors, regions, and margins to external files. Default to false.

Parameters

`pathname` Directory path to put the files.

Example usage:

```cpp
Sam sam("../data/toy1/SAM_USA_2004.csv");
sam.extractSFRM();
sam.writeLists("../data/toy1/out/");
```

The documentation for this class was generated from the following files:

- Sam.h
- Sam.cpp

4.14  SamDim Class Reference

This class contains the structure information of submatrices in a SAM.

```
#include <SamDim.h>
```

Public Member Functions

- **SamDim** (const int)
  
  *SamDim* constructor sets the dimensions of a *Sam*.

- **SamDim** (const double=7.1)
  
  *SamDim* constructor sets the dimensions of a *Sam*.

- **SamDim** (const int, const int, const int, const int, const int=1)
  
  *SamDim* constructor sets the dimensions of a *Sam*.

- virtual **~SamDim** ()
  
  *SamDim* default destructor.

- virtual int **getC** () const

---

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
4.14 SamDim Class Reference

Returns number of CGDS.
• virtual int getF () const
  Returns number of aggregated factors.
• virtual int getM () const
  Returns number of margin commodities.
• virtual int getR () const
  Returns number of aggregated regions.
• virtual int getS () const
  Returns number of aggregated sectors.
• virtual double getVersion () const
  Returns GTAP version number of SAM data.
• virtual int getYear () const
  Returns the year associated with SAM data.
• virtual size_t getI1 (const size_t) const
  Gets the index of the first row in a SAM sub-matrix S(I,:).
• virtual size_t getI2 (const size_t) const
  Gets the index of the last row in a SAM sub-matrix S(I,:).
• virtual size_t getJ1 (const size_t) const
  Gets the index of the first column in a SAM sub-matrix.
• virtual size_t getJ2 (const size_t) const
  Gets the index of the last column in a SAM sub-matrix S(:,J).
• virtual void computeIndices ()
  Compute indices of 19 x 19 submatrices in S.
• virtual std::string getHeader (const size_t) const
  Gets the header for a given column or row index of a SAM's submatrices.
• virtual size_t getSize () const
  Compute Sam's size from formula (5s+2f+3r+mr+m+l), l = 7 by default.
• virtual size_t getBlockNumber () const
  Compute Sam's size from formula (5s+2f+3r+mr+m+l), l = 7 by default.
• virtual void setC (const int)
  Sets number of CGDS.
• virtual void setF (const int)
  Sets number of aggregated factors.
• virtual void setM (const int)
  Sets number of margin commodities.
• virtual void setR (const int)
  Sets number of aggregated regions.
• virtual void setS (const int)
  Sets number of aggregated sectors.
• virtual void setSFRMC (const int, const int, const int, const int, const int=1)
Sets the dimensions of a Sam.

- virtual void setVersion (const double)
  Sets GTAP version number of SAM data.
- virtual void setYear (const int)
  Sets the year associated with SAM data.
- virtual void setDefaultCloseOperation ()
  Sets the default headers for a SAM's submatrices.
- virtual void setHeader (const size_t, const std::string)
  Sets the header for a given column or row of a SAM's submatrices.
- virtual void print () const
  Prints some field values of this SamDim instance.
- virtual size_t getBlockIndex (const size_t) const
  Get block index.
- virtual std::string getBlockHeader (const size_t, const size_t) const
  Get block header, i.e., summary description.
- bool operator== (const SamDim &samdim2) const
  Overload operator == for comparing two SamDim instances.
- bool operator!= (const SamDim &samdim2) const
  Overload operator != for comparing two SamDim instances.
- virtual int getL () const
- virtual void setL (int l=7)
  Sets value of L.

Static Public Member Functions

- static bool isRegion (const size_t)
  Checks if block index is associated with a region.
- static bool isSector (const size_t)
  Checks if block index is associated with a sector.
- static bool isFactor (const size_t)
  Checks if block index is associated with a factor.
- static bool isMargin (const size_t)
  Checks if block index is associated with margin commodities.

Static Public Attributes

- static const size_t T_length = 19
  Number of submatrices = 19 x 19.
- static const int sectorsSize = 5
4.14 SamDim Class Reference

Number of submatrices that correspond to sectors = 5.

- static const int sectorIndices[] = {1, 2, 3, 7, 8}

Indices of submatrices that correspond to sectors.

- static const int regionsSize = 4

Number of submatrices that correspond to regions = 4.

- static const int regionIndices[] = {5, 6, 10, 12}

Indices of submatrices that correspond to regions.

- static const int factorsSize = 2

Number of submatrices that correspond to factors = 2.

- static const int factorIndices[] = {4, 9}

Indices of submatrices that correspond to factors.

- static const int marginsSize = 1

Number of submatrices that correspond to margins = 1.

- static const int marginIndices[] = {11}

Indices of submatrices that correspond to margins.

- static const int last7Indices[] = {13, 14, 15, 16, 17, 18, 19}

Indices of submatrices that correspond to the last 7 rows/columns.

- static const int expectedPattern[19][19]

Expected nonzero blocks of submatrices.

- static const int isSumPositive[19]

Expected positive row old column block of submatrices.

4.14.1 Detailed Description

This class contains the structure information of submatrices in a SAM.

This class contains the structure information such as headers; version; numbers of sectors, regions, factors, margins; and dimensions of 19 times 19 submatrices in a standardized SAM.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 SamDim::SamDim (const int year)

SamDim constructor sets the dimensions of a Sam.

Parameters

| year  | Year of SAM data. Optional and defaults to 2004 (GTAP version 7.1). |

Generated on Thu Aug 29 2013 10:51:32 for OSCEF APIs by Doxygen
4.14.2.2  SamDim::SamDim ( const double version = 7.1 )

SamDim constructor sets the dimensions of a Sam.

Parameters

<table>
<thead>
<tr>
<th>version</th>
<th>GTAP version number. Optional and defaults to 7.1 (year 2004).</th>
</tr>
</thead>
</table>

4.14.2.3  SamDim::SamDim ( const int s, const int f, const int r, const int m, const int c = 1 )

SamDim constructor sets the dimensions of a Sam.

Parameters

| s          | Number of aggregated sectors. |
| f          | Number of aggregated factors. |
| r          | Number of aggregated regions. |
| m          | Number of margin commodities. |
| c          | Number of CGDS. Optional and default to 1. |

4.14.3  Member Function Documentation

4.14.3.1  void SamDim::computeIndices ( ) [virtual]

Compute indices of 19 x 19 submatrices in S.

See also

globIndex

4.14.3.2  std::string SamDim::getBlockHeader ( const size_t i, const size_t j ) const [virtual]

Get block header, i.e., summary description.

Parameters

| i         | Positive integer between 1 to 19. |
| j         | Positive integer between 1 to 19. |
Returns

A non-empty summary string if I and J are valid, and an empty string otherwise.

See also

generic

4.14.3.3 size_t SamDim::getBlockIndex ( const size_t i ) const [virtual]

Get block index.

Parameters

i Row or column index of a sam matrix, i=1,...,n, where n is size of SAM.

Returns

Block index between 1 to L. If it is -1, it means input i is out of the valid range of [1,n].

See also

calculateIndices, getL

4.14.3.4 std::string SamDim::getHeader ( const size_t j ) const [virtual]

Gets the header for a given column or row index of a SAM's submatrices.

Parameters

j Positive integer. Column or row block index.

Returns

Header of the j-th column or row block of S.

4.14.3.5 size_t SamDim::getI1 ( const size_t I ) const [virtual]

Gets the index of the first row in a SAM sub-matrix S(I,:).
Parameters

\[ I \mid \text{Positive integer between 1 to 19.} \]

Returns

\( I_1 \), index of the first row in the SAM sub-matrix such that \( T(I,:) = S(I_1:I_2,:) \).

4.14.3.6 size_t SamDim::getI2 ( const size_t I ) const [virtual]

 Gets the index of the last row in a SAM sub-matrix \( S(I,:) \).

Parameters

\[ I \mid \text{Positive integer between 1 to 19.} \]

Returns

\( I_2 \), index of the last row in the SAM sub-matrix such that \( T(I,:) = S(I_1:I_2,:) \).

4.14.3.7 size_t SamDim::getJ1 ( const size_t J ) const [virtual]

 Gets the index of the first column in a SAM sub-matrix.

Parameters

\[ J \mid \text{Positive integer between 1 to 19.} \]

Returns

\( J_1 \), index of the first column in the SAM sub-matrix such that \( T(:,J) = S(:,J_1:J_2) \).

See also

getJ2()

4.14.3.8 size_t SamDim::getJ2 ( const size_t J ) const [virtual]

 Gets the index of the last column in a SAM sub-matrix \( S(:,J) \).

Parameters

\[ J \mid \text{Positive integer between 1 to 19.} \]
4.14 SamDim Class Reference

Returns

J2, index of the last column in the SAM sub-matrix such that T(:,J) = S(:,J1:J2).

4.14.3.9 int SamDim::getL ( ) const [virtual]

Returns

L, which is the number of the last few rows (or columns) in a SAM that are neither sectors, factors, regions, margins, or commodities.

See also

setL

4.14.3.10 bool SamDim::isFactor ( const size_t I ) [static]

Checks if block index is associated with a factor.

Parameters

I | Positive integer between 1 to 19, representing a row or block index.

Returns

True if it is a factor block index, false otherwise.

4.14.3.11 bool SamDim::isMargin ( const size_t I ) [static]

Checks if block index is associated with margin commodities.

Parameters

I | Positive integer between 1 to 19, representing a row or block index.

Returns

True if it is a margin commodities block index, false otherwise.

4.14.3.12 bool SamDim::isRegion ( const size_t I ) [static]

Checks if block index is associated with a region.
Parameters

\( I \) | Positive integer between 1 to 19, representing a row or block index.

Returns

True if it is a region block index, false otherwise.

4.14.3.13 bool SamDim::isSector ( const size_t \( I \) ) [static]

Checks if block index is associated with a sector.

Parameters

\( I \) | Positive integer between 1 to 19, representing a row or block index.

Returns

True if it is a sector block index, false otherwise.

4.14.3.14 bool SamDim::operator!= ( const SamDim & samdim2 ) const

Overload operator \(!=\) for comparing two SamDim instances.

samdim1 Left-hand-side SamDim instance.

Parameters

samdim2 | Right-hand-side SamDim instance.

Returns

True if samdim1 and samdim2 do not have the same values of c, f, m, r, s, year, and version values.

4.14.3.15 bool SamDim::operator== ( const SamDim & samdim2 ) const

Overload operator \(==\) for comparing two SamDim instances.

samdim1 Left-hand-side SamDim instance.

Parameters

samdim2 | Right-hand-side SamDim instance.
4.14 SamDim Class Reference

Returns
True if samdim1 and samdim2 have the same values of c, f, m, r, s, year, and version values.

4.14.3.16 void SamDim::setHeader ( const size_t j, const std::string s ) [virtual]

Sets the header for a given column or row of a SAM's submatrices.

Parameters

<table>
<thead>
<tr>
<th>j</th>
<th>Positive integer. Column or row block index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Column or row block label.</td>
</tr>
</tbody>
</table>

4.14.3.17 void SamDim::setL ( int l = 7 ) [virtual]

Sets value of L.

Parameters

| l | The number of the last few rows (or columns) in a SAM that are neither sectors, factors, regions, margins, or commodities. Default to 7. |

See also
getL

4.14.3.18 void SamDim::setSFRMC ( const int s, const int f, const int r, const int m, const int c = 1 ) [virtual]

Sets the dimensions of a Sam.

Parameters

<table>
<thead>
<tr>
<th>s</th>
<th>Number of aggregated sectors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>Number of aggregated factors.</td>
</tr>
<tr>
<td>r</td>
<td>Number of aggregated regions.</td>
</tr>
<tr>
<td>m</td>
<td>Number of margin commodities.</td>
</tr>
<tr>
<td>c</td>
<td>Number of CGDS. Optional and default to 1.</td>
</tr>
</tbody>
</table>
4.14.3.19  void SamDim::setVersion ( const double version )  [virtual]

Sets GTAP version number of SAM data.

Parameters

| version | Version number of GTAP SAM data. |

4.14.3.20 void SamDim::setYear ( const int year )  [virtual]

Sets the year associated with SAM data.

Parameters

| year | Year information of GTAP SAM data. |

4.14.4  Member Data Documentation

4.14.4.1  const int SamDim::expectedPattern  [static]

Initial value:

```
{
  { 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 2},
  { 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 2, 0, 2},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
```

Expected nonzero blocks of submatrices.

An expectedPattern[i,j] is 0 if the submatrix could be zero, 1 if the submatrix could be nonzero, 2 if must be positive.
4.15 Sams Class Reference

4.14.2 const int SamDim::isSumPositive [static]

Initial value:

{2,1,1,2,1,
  1,1,1,1,2,
  0,2,1,1,0,
  1,1,2,1 }

Expected positive row old column block of submatrices.
An isSumPositive[i] is 0 if the submatrix could be zero, 1 if the submatrix could be nonzero, 2 if must be positive.

The documentation for this class was generated from the following files:

• SamDim.h
• SamDim.cpp

4.15 Sams Class Reference

Social accounting matrices in standardized dense or sparse formats.

#include <Sams.h>

Public Member Functions

• Sams ()
  Sams default constructor.
• Sams (const string, const bool=false, const bool=false, const bool=false, const double=7.1)
  Sams constructor reads one or more external standardized SAM files.
• virtual ~Sams ()
  Sams default destructor.
• virtual void read (const string, const bool=false, const bool=false, const bool=false, const double=7.1)
  Reads one or more external standardized SAM files.
• virtual void write (const string, const bool=false) const
  Writes one or multiple external standardized SAMs files.
• virtual size_t getSize () const
  Gets the number of SAM matrices.
• virtual Sam getSam (const string) const
  Gets a Sam matrix.
virtual Sam getSam (const int) const

  Gets a Sam matrix.

virtual void regionAggregate (const string=string())

  Aggregates all SAMS matrices according to the input region mapper.

virtual void regionAggregate (const Mapper &)

  Aggregates all SAMS matrices according to the input region mapper.

virtual void regionDisaggregate (const string=string())

  Proportionally disaggregates all SAMS matrices according to the input region mapper.

virtual void regionDisaggregate (const Mapper &)

  Proportionally disaggregates all SAMS matrices according to the input region mapper.

virtual void sectorAggregate (const string=string())

  Aggregates a Sam matrix according to the input sector mapper.

virtual void sectorAggregate (const Mapper &)

  Aggregates a Sam matrix according to the input sector mapper.

virtual void marginAggregate (const string=string())

  Aggregates a Sam matrix according to the input margin mapper.

virtual void marginAggregate (const Mapper &)

  Aggregates all Sam matrices according to the input margin mapper.

virtual void factorAggregate (const string=string())

  Aggregates a Sam matrix according to the input factor mapper.

virtual void factorAggregate (const Mapper &)

  Aggregates all Sam matrices according to the input factor mapper.

virtual void sectorDisaggregate (const string=string())

  Disaggregates a Sam matrix according to the input sector mapper.

virtual void sectorDisaggregate (const Mapper &)

  Disaggregates a Sam matrix according to the input sector mapper.

virtual bool operator== (const Sams &) const

  Overload operator == for comparing two Sams objects.

virtual bool operator!= (const Sams &) const

  Overload operator != for comparing two Sams objects.

virtual void setSam (const Sam &)

  Sets or adds a Sam matrix.

virtual void setSamDim (const SamDim &)

  Sets the defining sub-block dimensions of every Sam matrix in this Sams instance.

virtual const SamDim & getSamDim () const

  Gets a SamDim instance associated with every Sam matrix in this Sams.

virtual bool areBalanced (const double=0) const

  Test if very Sam matrix in this Sams is balanced.

virtual void setOutOfCore (const bool)
4.15 Sams Class Reference

4.15.1 Detailed Description

Social accounting matrices in standardized dense or sparse formats. Social accounting matrices in standardized dense or sparse formats with properties including size, region, year, row sums, column sums, matrix sums, number and percentage of non-zeros.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Sams::Sams (const string path, const bool sparse = false, const bool sparseFormat = false, const bool outOfCore = false, const double version = 7.1)

Sams constructor reads one or more external standardized SAM files.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to a directory that contains one or more SAM files.</td>
</tr>
<tr>
<td>sparse</td>
<td>Flag to use sparse data structure. Default to false.</td>
</tr>
<tr>
<td>sparseFormat</td>
<td>Flag to indicate that the input file(s) are in sparse-matrix format. Default to false.</td>
</tr>
<tr>
<td>outOfCore</td>
<td>Flag to read each input file out of core. Default to false.</td>
</tr>
<tr>
<td>version</td>
<td>Version number of GTAP SAM data. Default to 7.1.</td>
</tr>
</tbody>
</table>


4.15.3 Member Function Documentation

4.15.3.1 bool Sams::areBalanced ( const double tol = 0 ) const [virtual]

Test if very Sam matrix in this Sams is balanced.

Parameters

| tol | Small positive number. Default to machine precision |

Returns

True if each row sum equals its corresponding column sum with a precision up to about the most significant \( \log_{10}(tol) \) digits.

Note

Consider calling computeSums() before calling this method.

4.15.3.2 void Sams::balanceByRas ( const Double_Vector & new_sums )

[protected, virtual]

Balance each Sam matrix in this Sams by iterative row and column scalings.

Parameters

| new_sums | Optional. Vector of new row and column sums. Default to a vector of size 0, in which case the algorithm will work to find a new balanced Sam matrix with row and column sums equal to the average of row and column sums of the Sam matrix. Otherwise, the algorithm will work to find a new balanced Sam matrix with row and column sums equal to new_sums. |

Returns

A balanced Sam matrix, or an empty sam matrix of size 0 if balancing operation could not be performed due to for example, the Sam matrix is empty, or if the size of new_sums is not equal to the size of the Sam matrix.
4.15 Sams Class Reference

4.15.3.3 void Sams::factorAggregate ( const string mapper_path = string() ) [virtual]

Aggregates a Sam matrix according to the input factor mapper.

Parameters

| mapper_path | Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance. |

Returns

this SAM, which has the aggregated factors, or becomes of size 0 if there is no SAM factor to be aggregated.

4.15.3.4 void Sams::factorAggregate ( const Mapper & mapper ) [virtual]

Aggregates all Sam matrices according to the input factor mapper.

Parameters

| mapper | A mapper instance. Optional. Defaults to an empty string, which returns the same Sams instance. |

Returns

this SAMs, which has the aggregated factors, or becomes of size 0 if there is no SAM factor to be aggregated.

4.15.3.5 bool Sams::getOutOfCore ( ) const [virtual]

Gets the value of whether each SAM is to be read or processed out of core.

Returns

True if each SAM is processed out of core.

4.15.3.6 Sam Sams::getSam ( const string code ) const [virtual]

Gets a Sam matrix.
### Parameters

| code         | Code of the Sam matrix to be retrieved. |

### Returns

A Sam matrix.

#### 4.15.3.7 Sam Sams::getSam ( const int index ) const [virtual]

Gets a Sam matrix.

### Parameters

| index       | Positive integer. Index of the region code of the Sam matrix to be retrieved. |

### Returns

A Sam matrix.

#### 4.15.3.8 const SamDim & Sams::getSamDim ( ) const [virtual]

Gets a SamDim instance associated with every Sam matrix in this Sams.

### Returns

A SamDim instance that defines the submatrix structure of all Sams.

#### 4.15.3.9 size_t Sams::getSize ( ) const [virtual]

Gets the number of SAM matrices.

### Returns

size of SAMs.

#### 4.15.3.10 bool Sams::isValid ( const double tol = util::EPS, const double drop = 0 ) [virtual]

Checks if every SAM in this SAMS object has valid values.
4.15 Sams Class Reference

Parameters

<table>
<thead>
<tr>
<th>tol</th>
<th>Relative tolerance for testing if two numbers are equal in the method Sam::isMatSumsValid(). Optional. Default to machine precision.</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop</td>
<td>If the magnitude of one or both numbers is less than the drop, in Sam::isMatSumsValid(), we do not hold them to the relative-tolerance test. Optional. Default to 0.</td>
</tr>
</tbody>
</table>

Returns

True if every SAM in this SAMS object has all values valid.

4.15.3.11 void Sams::marginAggregate ( const string mapper_path = string() ) [virtual]

Aggregates a Sam matrix according to the input margin mapper.

Parameters

mapper_path | Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance. |

Returns
this SAM, which has the aggregated margins, or becomes of size 0 if there is no SAM margin to be aggregated.

4.15.3.12 void Sams::marginAggregate ( const Mapper & mapper ) [virtual]

Aggregates all Sam matrices according to the input margin mapper.

Parameters

mapper | A mapper instance. Optional. Defaults to an empty string, which returns the same Sams instance. |

Returns
	his SAMs, which has the aggregated magins, or becomes of size 0 if there is no SAM factor to be aggregated.
4.15.3.13 bool Sams::operator!= ( const Sams & sams2 ) const [virtual]

Overload operator != for comparing two Sams objects.
sams1 Left-hand-side SAM matrix.

Parameters

- **sams2** Right-hand-side SAM matrix.

Returns

False if sams1 and sams2 have the same size and contain Sam matrices of the same codes and values.

4.15.3.14 bool Sams::operator== ( const Sams & sams2 ) const [virtual]

Overload operator == for comparing two Sams objects.
sams1 Left-hand-side SAM matrix.

Parameters

- **sams2** Right-hand-side SAM matrix.

Returns

True if sams1 and sams2 have the same size and contain Sam matrices of the same codes and values.

4.15.3.15 void Sams::read ( const string path, const bool sparse = false, const bool sparseFormat = false, const bool outOfCore = false, const double version = 7.1 ) [virtual]

Reads one or more external standardized SAM files.

Parameters

<table>
<thead>
<tr>
<th><strong>path</strong></th>
<th>Path to a directory that contains one or more SAM files. Note that it is case sensitive.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sparse</strong></td>
<td>Flag to use sparse data structure.</td>
</tr>
<tr>
<td><strong>sparseFormat</strong></td>
<td>Flag to indicate that the input file(s) are in sparse-matrix format. Default to false.</td>
</tr>
<tr>
<td><strong>outOfCore</strong></td>
<td>Flag to read each input file out of core.</td>
</tr>
<tr>
<td><strong>version</strong></td>
<td>Version number of GTAP SAM data. Default to 7.1.</td>
</tr>
</tbody>
</table>
4.15 Sams Class Reference

Note

This method will not recursively read into subdirectories.

4.15.3.16 void Sams::regionAggregate ( const string mapper_path = string() )
                   [virtual]

Aggregates all SAMS matrices according to the input region mapper.

Parameters

| mapper_path | path to a mapper file. Optional. Defaults to an empty string, in which case the whole sams is intact. |

Returns

a Sams instance, which is the sum of all SAMs; or a SAM of size 0 if there is no SAM to be summed.

4.15.3.17 void Sams::regionAggregate ( const Mapper & mapper ) [virtual]

Aggregates all SAMS matrices according to the input region mapper.

Parameters

| mapper | A Mapper instance. |

Returns

a Sams instance with SAMs aggregated at Sam-file and region-column level; or a SAM of size 0 if there is no SAM to be summed.

4.15.3.18 void Sams::regionDisaggregate ( const string mapper_path = string() )
                   [virtual]

Proportionally disaggregates all SAMS matrices according to the input region mapper.

Parameters

| mapper_path | path to a mapper file. |
Returns a `Sams` instance with disaggregated `Sam` matrices; or a SAM of size 0 if there is no `Sam` matrix to be disaggregated.

Note

Given an input region code and its associated m output region code(s), m a positive integer, proportionally disaggregation means the each of the output-region `Sam` is set to be input-region `Sam` divided by m.

4.15.3.19 `void Sams::regionDisaggregate ( const Mapper & mapper ) [virtual]`

Proportionally disaggregates all SAMS matrices according to the input region mapper.

Parameters

- `mapper` A mapper instance.

Returns a `Sams` instance with disaggregated `Sam` matrices; or a SAM of size 0 if there is no `Sam` matrix to be disaggregated.

Note

Given an input region code and its associated m output region code(s), m a positive integer, proportionally disaggregation means the each of the output-region `Sam` is set to be input-region `Sam` divided by m.

4.15.3.20 `void Sams::sectorAggregate ( const string mapper_path = string() ) [virtual]`

Aggregates a `Sam` matrix according to the input sector mapper.

Parameters

- `mapper_path` Path to a mapper file. Optional. Defaults to an empty string, which returns the same sam instance.
4.15 Sams Class Reference

Returns

this SAM, which has the aggregated sectors, or becomes of size 0 if there is no
SAM sector to be aggregated.

4.15.3.21 void Sams::sectorAggregate ( const Mapper & mapper ) [virtual]

Aggregates a Sam matrix according to the input sector mapper.

Parameters

mapper A mapper instance. Optional. Defaults to an empty string, which returns
the same sam instance.

Returns

this SAM, which has the aggregated sectors, or becomes of size 0 if there is no
SAM sector to be aggregated.

4.15.3.22 void Sams::sectorDisaggregate ( const string mapper_path = string() )
[virtual]

Disaggregates a Sam matrix according to the input sector mapper.

Parameters

mapper_ path Path to a mapper file. Optional. Defaults to an empty string, which
returns the same sam instance.

Returns

this SAM, which has the disaggregated sectors, or becomes of size 0 if there is no
SAM sector to be disaggregated.

4.15.3.23 void Sams::sectorDisaggregate ( const Mapper & mapper ) [virtual]

Disaggregates a Sam matrix according to the input sector mapper.

Parameters

mapper A mapper instance. Optional. Defaults to an empty string, which returns
the same sam instance.
Returns

this SAM, which has the disaggregated sectors, or becomes of size 0 if there is no SAM sector to be disaggregated.

### 4.15.3.24 void Sams::setSam ( const Sam & sam ) [virtual]

Sets or adds a Sam matrix.

**Parameters**

| sam | The Sam instance to be set or added (if it did not exist before). |

### 4.15.3.25 void Sams::setSamDim ( const SamDim & samDim ) [virtual]

Sets the defining sub-block dimensions of every Sam matrix in this Sams instance.

**Parameters**

| samDim | A SamDim instance. |

### 4.15.3.26 void Sams::write ( const string path, const bool sparseFormat = false ) const [virtual]

Writes one or multiple external standardized SAMs files.

**Parameters**

| path | Path to write one or multiple SAMs files. | sparse-Format | Flag to indicate that the output file(s) will be written in sparse-matrix format. Default to false. |

The documentation for this class was generated from the following files:

- Sams.h
- Sams.cpp

### 4.16 Sectors Class Reference

A number-indexed list of sectors.

```
#include <Sectors.h>
```
Inheritance diagram for Sectors:

```
List
  ^
  v
Sectors
```

Public Member Functions

- `Sectors ()`
  
  *Sectors* default constructor.

- `Sectors (const string)`
  
  *Sectors* constructor reads an external standardized regions file.

- `virtual ~Sectors ()`
  
  *Sectors* default destructor.

### 4.16.1 Detailed Description

A number-indexed list of sectors.
A numbered list of codes that represent sectors.

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 Sectors::Sectors ( const string *filename* )

*Sectors* constructor reads an external standardized regions file.

**Parameters**

```
filename       path to a regions file.
```

The documentation for this class was generated from the following files:

- Sectors.h
- Sectors.cpp
4.17 Utilities Class Reference

Utilities or tools as inline functions.

```c
#include <Utilities.h>
```

4.17.1 Detailed Description

Utilities or tools as inline functions.

Utilities or tools as inline functions for processing and extracting substrings, and comparing real numbers.

The documentation for this class was generated from the following file:

- Utilities.h
5.1 Check

Checks if two double numbers are equal with a small prescribed error.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>First double number.</td>
</tr>
<tr>
<td>b</td>
<td>Second double number.</td>
</tr>
<tr>
<td>tol</td>
<td>Positive real number. Tolerance for error.</td>
</tr>
<tr>
<td>drop</td>
<td>Nonnegative real number. Minimum magnitude of a to be checked for relative tolerance. Default to 0.</td>
</tr>
</tbody>
</table>

Returns

True if the magnitude of the difference between the two numbers is less than or equal to the magnitude of delta.

if the following relative difference is true: \( \frac{\text{abs}(a-b)}{\max(\text{abs}(a), \text{drop})} > \text{tol} \)