Gurobi Python 矩阵建模操作

仅限 Gurobi 10.0 版本或以上

仅限 Python 3.7 版本或以上

help@gurobi.cn

更新日期 2022-11-10

准备知识

从版本 10.0 开始，Gurobi Python API 升级了对矩阵操作的支持。通过创建任意维度矩阵变量、矩阵约束、矩阵线性表达式、矩阵二次表达式等，让用户方便、快捷地进行矩阵运算，并且可以和常规变量、约束等混合使用，提升了建模效率，更易学易用。

Gurobi 矩阵建模建立在 Numpy ndarray 和 Scipy.sparse 基础之上。

关于 Numpy ndarray 的基础知识参见
https://numpy.org/doc/stable/user/absolute_beginners.html
和
https://numpy.org/doc/stable/user/quickstart.html

关于 Scipy.sparse 的基础知识参见
https://docs.scipy.org/doc/scipy/reference/sparse.html

关于Gurobi 矩阵操作的范例参见
Gurobi安装目录/examples/python 下的matrx1.py, matrix2.py

矩阵操作的基本知识

维度（Dimension）和形状（Shape）

利用 ndarray.ndim 和 ndarray.shape 输出任何矩阵（含向量、标量）的维度和形状信息。用 ndarray.T 获得转置后矩阵。和 Gurobi 建模相关的有如下信息：

操作	输出	维度	形状	转置后形状	常用用途
A = numpy.array([[1,2,3,4],[5,6,7,8]])	[[1 2 3 4] [5 6 7 8]]	2-D	(2,4)	(4,2)	系数矩阵、矩阵变量和表达式
B = A[0:1, :]	[[1 2 3 4]]	2-D	(1,4)	(4,1)	行（列）向量变量或者系数
C = A[0, :]	[1 2 3 4]	1-D	(4,)	(4,) 保持不变	一维变量或者系数
D= A[0,2]	3	0-D	( )	( ) 保持不变	标量变量或者常数

Gurobi MVar 矩阵变量

Gurobi Mvar 矩阵变量

Gurobi Mvar 矩阵变量是 Numpy ndarray 类型，维度和形状取决于定义时的输入参数。

操作	维度	形状	转置后形状	说明
x = model.addMVar((2,4))	2-D	(2,4)	(4,2)	MVar 变量
y = x[0:1, :]	2-D	(1,4)	(4,1)	行（列）变量
z = x[0, :]	1-D	(4,)	(4,) 保持不变	一维变量
w= x[0,2]	0-D	( )	( ) 保持不变	标量变量，等同于常规通过 model.addVar() 创建的单变量

注意: y,z,w 都不是新变量，只是 x 变量的一部分，属性沿用 x 变量属性（包括 name)

Gurobi MVar 矩阵变量常用函数和属性

MVar.copy()
MVar.diagonal()
MVar.fromlist()
MVar.getAttr()
MVar.item()
MVar.ndim
MVar.reshape()
MVar.setAttr()
MVar.shape
MVar.size
MVar.sum()
MVar.T
MVar.tolist()
MVar.transpose()

Gurobi MVar 矩阵变量常用函数和属性

MVar.copy()

返回 MVar 的拷贝，变量没有增加

举例：

```
orig = model.addMVar(3)
copy = orig.copy()
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.diagonal(offset=0, axis1=0, axis2=1)

返回指定对角线上的 MVar 变量

Offset: 相对于主对角线的便宜，>0 指主对角线之上, <0 指主对角线之下
axis1: 对于超过 2-D Mvar 指定决定子 2-D Mvar的第一个轴线
axis2: 对于超过 2-D Mvar 指定决定子 2-D Mvar的第二个轴线

举例：

```
x = model.addMVar((8, 8))
diag_main = x.diagonal()  # 1-D MVar 包含 x 主对角线变量
diag_sup = x.diagonal(1)  # 1-D MVar 包含 x 主对角线向上偏离一位的变量
diag_sub = x.diagonal(-2)  # 1-D MVar 包含 x 主对角线向下偏离二位的变量
adiag_main = x[:, ::-1].diagonal()  # 1-D Mvar 包含 x的 反对角线变量
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.fromlist()

从一列或二列常规变量转换为1-D 或者 2-D Mvar，转换后的 MVar 只是常规变量的新组合，并未增加新变量。

举例

```
x0 = model.addVar()
x1 = model.addVar()
x2 = model.addVar()
x3 = model.addVar()
x_1d = gurobipy.MVar.fromlist([x0, x1, x2, x3])  # 1-D MVar
x_2d = gurobipy.MVar.fromlist([[x0, x1], [x2, x3]])  # 2-D MVar
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.getAttr(attrname)

获得 MVar 的变量属性值，以 Numpy ndarray 结构返回，形状和 MVar 一致

举例：

```
print(var.getAttr(GRB.Attr.X))
print(var.getAttr("x"))
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.item()

从只包含1个元素的 Mvar 中获取常规变量。当 Mvar 包含超过1个元素时会报错

举例

```
x = model.addMVar((2, 2))
x_sub = x[0, 1]  #  0-D MVar 包含了一个 Var 对象
x_var = x[0, 1].item()  #  Var 对象本身
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.ndim： 返回 MVar 的维度数量

MVar.shape：返回 MVar 的形状

MVar.size：返回 MVar 包含的元素数量

MVar.T：返回 MVar 的转置

MVar.transpose()：返回 MVar 的转置，等同于 MVar.T

Gurobi MVar 矩阵变量常用函数和属性

MVar.reshape(shape, order='C')

重组 MVar 的元素到新的形状 Mvar变量

Shape: 新的形状。如果某个维度数值为-1，则意味着这个维度的数值将由总元素数量和其他维度的数值来确定。

order: 取值 'C' 或者 'F', 决定MVar 元素是以何种次序读取并重组。

举例

```
x = model.addMVar((2, 2))
x_row = x.reshape(-1, order='C')  # 1-D MVar, 按行读取
x_col = x.reshape(-1, order='F')  # 1-D MVar, 按列读取
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.setAttr(attrname, newvalue)

设置 MVar 变量的属性值。设置后只有在模型更新后（update,optimize,write)才会生效

attrname：属性名称
newvalue: 新属性值，形状需要和 Mvar 一致，或者是一个标量

举例

```
MVar.setAttr("ub", np.full((5,), 10))
MVar.setAttr(GRB.Attr.UB, 10.0)
MVar.setAttr("ub", 10.0)
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.sum(axis=None)

对 MVar 里的变量进行加和，返回 MLinExpr 对象。

Axis: 沿指定轴进行加和。不指定时加和全部MVar元素

举例

```
x = model.addMVar((2, 2))
sum_row = x.sum(axis=0)  # 按照列加和，返回形状 (2,)
sum_col = x.sum(axis=1)  # 按照行加和，返回形状 (2,)
sum_all = x.sum()  # 所有元素加和
```

Gurobi MVar 矩阵变量常用函数和属性

MVar.tolist()

将 MVar 变量转换为常规变量列表。变量只是重组，没有增加。

举例

```
mvar = model.addMVar(5)
varlist = mvar.tolist()
print(varlist)
```

Gurobi MLinExpr 矩阵线性表达式

Gurobi MLinExpr 矩阵线性表达式

通过 MVar 变量和矩阵（向量）系数（ NumPy ndarray 或者 SciPy sparse matrix 类型）组合在一起的线性表达式。

- 最常见的表达方式：expr = A @ x 其中，A 是 2-D 矩阵，x 是 1-D MVar 变量
- 加常数 expr = x + 1 / 乘常数 expr = 2 * A @ x
- 加减法 expr = A @ x - B @ y
- 支持 += / -= / *=
- MLinExpr.shape 输出形状； MLinExpr.size 输出包含的元素个数; MLinExpr.ndim 输出维度数量。如果 MLinExpr 的形状是()，那么MLinExpr 代表标量，size为1，ndim 为0
- 多个 MLinExpr 对象组合在一起时，注意形状匹配。二个矩阵（向量）相乘（@乘法），相邻维度需要相同。二个矩阵（向量）点乘时（* 乘法），形状需要相同。
- MLinExpr和常规变量/LinExpr 混用时，MLinExpr 的形状需要是()，最终的表达式是 MLinExpr 类型，形状为()
- 多个 MLinExpr 组合时，遵循 Numpy的 broadcast （传播）规则（请参考 Numpy相关文献）

Gurobi MLinExpr 矩阵线性表达式（举例）

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np
import scipy.sparse as sp

model=gp.Model()
mx = model.addMVar((3,4))        # 2-D MVar 形状 (3,4)
my = mx[:, 0]                    # 1-D Mvar 形状 (3,)
c=np.array([1,2,3])
b=np.array([4,5,6,7])
mz = c @ mx                      # 1-D MLinExpr 形状(4,)，注意 mx @ c 形状不匹配会报错
x = model.addVar()
z = model.addVar()
mle = 2*x + c @ my               # 0-D MLinExpr 形状()
model.addConstr(z <=mle)
model.addConstr(mz >= b)
model.write('mlinexpr.lp')
```

Gurobi MLinExpr 矩阵线性表达式 常用函数和属性

MLinExpr.clear()

MLinExpr.copy()

MLinExpr.getValue()

MLinExpr.item()

MLinExpr.ndim

MLinExpr.shape

MLinExpr.size

MLinExpr.sum()

MLinExpr.zeros()

MLinExpr 索引和切片（读）

MLinExpr 索引和切片（写）

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.clear()

重置 MLinExpr 到全部为0

举例：

```
expr = 2 * model.addMVar(3) + 1
expr.clear() # 所有三个元素都重置为 0.0
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.copy()

创建 MLinExpr 的拷贝

举例：

```
orig = 2 * model.addMVar(3) + 1.0
copy = orig.copy()
copy += 2.0   # 原来的 orig 没有变化
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.getValue()

用当前的优化结果计算 MLinExpr 的数值

举例：

```
expr = A @ x + b
model.addConstr(expr == 0)
model.optimize()
val = expr.getValue()     # val 的形状和 expr 的形状一致
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.item()

对于只包含1个元素的 MLinExpr，调用 item()可以返回该元素的拷贝，类型为LinExpr

举例：

```
mle = 2 * model.addMVar((2, 2)) + 1
mle_sub = mle[0, 1]  # 0-D MLinExpr 只包含一个 LinExpr 对象
mle_le = mle[0, 1].item()  # 该 LinExpr对象的拷贝。修改 mle_le 不影响 mle
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.ndim

返回 MLinExpr 的维度

MLinExpr.shape

返回 MLinExpr 的形状

MLinExpr.size

返回 MLinExpr 的元素个数

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr.sum(axis=None**)**

加和MLinExpr 中的元素。

参数：
axis: 整数，指定加和的维度。如果没有指定，则全部元素相加。

举例：

```
expr = 2 * model.addMVar((2, 4)) - 1
sum_row = expr.sum(axis=0)  # 按照列加和，返回形状(4,)
sum_col = expr.sum(axis=1)  # 按照行加和，返回形状(2,)
sum_all = expr.sum()  # 所有元素加和，返回形状()
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr 索引和切片（读）

可以对 MLinExpr 正常进行索引和切片操作，正如对 Numpy ndarray 操作一样。需要注意的是，如果存在对于索引和切片进行写入操作的话，可能会影响到原始 MLinExpr 对象。

举例：

```
mle = 2 * m.addMVar(4)
leading_part_1 = mle[:2]
leading_part_2 = mle[[0,1]]
leading_part_1 += 99  # 同样修改了 mle
leading_part_2 += 1  # 没有修改 mle
```

如果不确定是否会修改，可以采用 copy() 函数

```
expr = 2 * model.addMVar((2,2)) + 1
first_col = expr[:, 0].copy()
first_col =+ 1  #  expr 没有变化
```

Gurobi MLinExpr 矩阵线性表达式 常用函数

MLinExpr 索引和切片（写）

对 MLinExpr 的索引和切片进行赋值操作。对 MLinExpr 的赋值操作因为涉及到多次数据拷贝，因此效率不如直接用 MVar 创建 MLinExpr 高。

举例：

```
mle = 2 * model.addMVar((2,2))
v = model.addVar()
w = model.addVar()
mle[:] = v + 1  # 覆盖写入 mle 4个元素均为 v+1 独立拷贝
mle[1, 1] = w  # 覆盖写入 mle  第 (1, 1) 元素为 w
```

Gurobi MQuadExpr 矩阵二次表达式

Gurobi MQuadExpr 矩阵二次表达式

通过 MVar 变量和矩阵（向量）系数（ NumPy ndarray 或者 SciPy sparse matrix 类型）组合在一起的二次表达式。

- 最常见的表达方式：expr = x @ A @ x 或者 expr = x @ x , 其中 A 是 2-D 矩阵，x 是 1-D MVar 变量
- 乘常数 expr = 2 * x @ A @ y ， 加减法 expr = x @ A @ x – y @ B @ y
- 支持 += / -= / *=
- MQuadExpr.shape 输出形状； MQuadExpr.size 输出包含的元素个数; MQuadExpr.ndim 输出 维度数量。如果 MQuadExpr 的形状是()，那么MQuadExpr 代表标量，size为1，ndim 为0
- 其他规则参考 MLinExpr

Gurobi MQuadExpr 矩阵二次表达式常用函数和属性

MQuadExpr.clear()
MQuadExpr.copy()
MQuadExpr.getValue()
MQuadExpr.item()
MQuadExpr.ndim
MQuadExpr.shape
MQuadExpr.size
MQuadExpr.sum()
MQuadExpr.zeros()
MQuadExpr 索引和切片（读）
MQuadExpr 索引和切片（写）

请参考 MLinExpr 用法和参考手册（安装目录/docs/refman.pdf）

Gurobi MConstr 矩阵约束

Gurobi MConstr 矩阵约束

- 仅限线性约束
- 包含一组由 MVar 和 MLinExpr 组合的线性约束
- 和 Numpy ndarray 处理类似，也有 shape, ndim 等属性，也可以进行索引和切片
- 和其他约束一样，Gurobi 采用惰性更新方式（Lazy Update）
- 通过 model.addMConstr() 或者 model.addConstr() 添加到模型中
- 可以通过 MConstr.Attr 对 MConstr 中的约束进行批量属性读取和修改

Gurobi MConstr 矩阵约束

MConstr.tolist()

MConstr.getAttr()

MConstr.setAttr()

Gurobi MConstr 矩阵约束

MConstr.tolist()

将Mconstr 中的约束转换为常规约束列表

举例

```
mc = model.addConstr(A @ x <= b)
constrlist = mc.tolist()
print(constrlist)
```

Gurobi MConstr 矩阵约束

MConstr.getAttr(attrname)

返回MConstr 中每个约束的属性值。返回结果是 Numpy ndarray， 形状和MConstr 一致

举例

```
mc = model.addConstr(A @ x <= b)
rhs = mc.getAttr("RHS")  # 也可以直接写为 rhs =  mc.RHS
```

Gurobi MConstr 矩阵约束

MConstr.setAttr(attrname, newvalue)

设置 MConstr的属性值。设置后只有在模型更新后（update,optimize,write)才会生效

attrname：属性名称
newvalue: 新属性值，形状需要和 MConstr 一致，或者是一个标量

举例

```
mc = model.addConstr(A @ x <= b)
mc.setAttr("RHS", np.arange(A.shape[0]))  # 也可以写为 mc.RHS = np.arange(A.shape[0])
mc.setAttr(GRB.Attr.RHS, 0.0)
```

Gurobi MQConstr 矩阵约束

Gurobi MQConstr 矩阵约束

- 二次约束
- 包含一组由 MVar 和 MQuadExpr 组合的二次约束
- 和 Numpy ndarray 处理类似，也有 shape, ndim 等属性，也可以进行索引和切片
- 和其他约束一样，Gurobi 采用惰性更新方式（Lazy Update）
- 通过 model.addMQConstr() 或者 model.addConstr() 添加到模型中
- 可以通过 MQConstr.Attr 对 MQConstr 中的约束进行批量属性读取和修改

Gurobi MQConstr 矩阵约束

MQConstr.tolist()

MQConstr.getAttr()

MQConstr.setAttr()

使用方法请参考 MConstr 对象

感谢对Gurobi的兴趣

完整信息见参考手册（Gurobi 安装目录/docs/refman.pdf）

范例见Gurobi 安装目录/examples/python 目录

更多问题可以加入 Gurobi QQ 群，群号见 http://www.gurobi.cn/about.asp?id=2