LinearSolver

August 4, 2024

1 Linear solvers

We demonstrate solving a dense system A.x=b by using different linear solvers.

[1]:
```
from casadi import *
from numpy import *
import time
```

[2]:
```
n=100
```

We generate $A \in \mathbf{R}^{n \times n}$, $x \in \mathbf{R}^n$ with $n = 100$

[3]:
```
A=DM([[cos(i*j)-sin(i) for i in range(n)] for j in range(n)])
x=DM([tan(i) for i in range(n)])
```

We generate the b vector:

[4]:
```
b= mtimes(A,x)
```

Commented out pendling completion #1615

2 We demonstrate the LinearSolver API with CSparse:

s = LinearSolver(“s”, “csparse”, A.sparsity())

3 Give it the matrix A

s.setInput(A,“A”) # Do the LU factorization s.prepare()

4 Give it the matrix b

s.setInput(b,“B”)

5 And we are off to find x...

s.solve()

x_ = s.getOutput(“X”)

6 By looking at the residuals between the x we knew in advance and the computed x, we see that the CSparse solver works

print “Sum of residuals = %.2e” % norm_1(x-x_)

7 Comparison of different linear solvers

8 ==========

for solver in (“lapacklu”,“lapackqr”,“csparse”): s = LinearSolver(“s”, solver, A.sparsity()) # We create a solver

s.setInput(A,“A”) # Give it the matrix A

t0 = time.time() for i in range(100): s.prepare() # Do the LU factorization pt = (time.time()-t0)/100

s.setInput(b,“B”) # Give it the matrix b

t0 = time.time() for i in range(100): s.solve() st = (time.time()-t0)/100

x_ = s.getOutput(“X”)

print ”” print solver print “=” * 10 print “Sum of residuals = %.2e” % norm_1(x-x_) print “Preparation time = %0.2f ms” % (pt*1000) print “Solve time = %0.2f ms” % (st*1000) assert(norm_1(x-x_)<1e-9)

9 Note that these