Verifying the bounds from `MUSSV`

Through elementary, but detailed calculations, the lower and upper bounds from `mussv` can be verified.

UC Berkeley, ME C231B/EECS C220C, Spring 2017

Contents

- Create block structure and compute `mussv` of a random matrix
- Check perturbation and lower bound
- Verify upper bound, using certificates from `VSigma`
- Verify upper bound, using certificates from `VLmi`
- Verfication for only square, complex blocks is easier
- Check perturbation and lower bound
- Verify upper bound, using certificates from `VSigma`
- Verify upper bound, using certificates from `VLmi`
- Upper-Bound quality
- Ordering of `MU` based on order of `DELTA` set
- Conclusions
- File Information
- Lessons learned from the file

Create block structure and compute `mussv` of a random matrix

```
blk = [-2 0;2 0];  % 1-by-1 real, repeated twice; 1-by-1 complex, repeated twice
M = complex(randn(4),randn(4));
[bnds,muinfo] = mussv(M,blk);
```

Check that upper bound is less than or equal to norm(M)

```
bnds(1)<=norm(M)
```

```
ans =

  logical

   1
```

Check that lower bound is greater than all real eigenvalues

```
eVal = eig(M);
realeVal = eVal(imag(eVal)==0);
max([realeVal;0])<=bnds(2)
```

```
ans =

  logical

   1
```

Extract verification parameters

```
[VDelta,VSigma,VLmi] = mussvextract(muinfo);
```

Check perturbation and lower bound

Verify that Delta has correct dimension

```
size(VDelta)
```

```
ans =

     4     4
```

Examine Delta, and confirm it has the proper structure.

```
VDelta
```

```
VDelta =

   0.3925 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.3925 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.3925 - 0.0049i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.3925 - 0.0049i
```

M*VDelta should have an eigenvalue at 1 (so det(I-M*VDelta)=0)

```
eig(M*VDelta)
```

```
ans =

   1.0000 - 0.0000i
   0.0943 + 0.6760i
   0.3960 + 0.4288i
  -0.6243 - 0.0398i
```

Lower bound should be equal to 1/norm(VDelta)

```
[bnds(2) 1/norm(VDelta)]
```

```
ans =

    2.5477    2.5477
```

Verify upper bound, using certificates from `VSigma`

Fields `GLeft`, `GMiddle` and `GRight` should be diagonal, with nonzero real entries associated with **real** uncertainties. These matrices should be "equal" although they may be different dimensions when `M` is nonsquare.

```
VSigma.GLeft
```

```
ans =

    0.8698         0         0         0
         0    0.2061         0         0
         0         0         0         0
         0         0         0         0
```

```
VSigma.GMiddle
```

```
ans =

    0.8698         0         0         0
         0    0.2061         0         0
         0         0         0         0
         0         0         0         0
```

```
VSigma.GRight
```

```
ans =

    0.8698         0         0         0
         0    0.2061         0         0
         0         0         0         0
         0         0         0         0
```

Fields `DLeft` and `DRight` should be block-diagonal, and structured consistent with the structure defined by `blk`. Specifically, for any matrix `Delta` with the structure defined by `blk`, it should be that `Delta*DLeft = DRight*Delta`. Both `DLeft` and `DRight` should be square and invertible, and "essentially" equal, although dimensions can be different. If `blk` contains any nonsquare full blocks, then the dimensions of `DLeft` and `DRight` will be different.

```
VSigma.DLeft
```

```
ans =

   0.0895 + 0.0017i  -0.0154 - 0.0043i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0098 - 0.0115i   0.0494 + 0.0015i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0418 + 0.0000i  -0.0040 + 0.0009i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0040 - 0.0009i   0.0551 + 0.0000i
```

```
VSigma.DRight
```

```
ans =

   0.0895 + 0.0017i  -0.0154 - 0.0043i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0098 - 0.0115i   0.0494 + 0.0015i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0418 + 0.0000i  -0.0040 + 0.0009i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0040 - 0.0009i   0.0551 + 0.0000i
```

The upper bound is verified by a norm test on a matrix involving the matrix `M`, the upper bound (`bnds(1)`, and the scalings. Form the matrix in 3 steps.

```
TL = (eye(4)+VSigma.GLeft^2)^-0.25;
TM = (1/bnds(1))*VSigma.DLeft*M/VSigma.DRight - sqrt(-1)*VSigma.GMiddle;
TR = (eye(4)+VSigma.GRight^2)^-0.25;
```

The upper bound is certified if NORM(TL*TM*TR)<=1. Check this

```
norm(TL*TM*TR)
```

```
ans =

    1.0000
```

Verify upper bound, using certificates from `VLmi`

Fields `Grc`, `Gcr` should be block-diagonal, with nonzero square matrix component entries associated with **real** uncertainties. These square block entries should be Hermitian, and the entries in `Grc` should equal those in `Gcr`.

```
VLmi.Gcr
```

```
ans =

   0.0135 - 0.0000i  -0.0026 - 0.0003i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0026 + 0.0003i   0.0017 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
VLmi.Grc
```

```
ans =

   0.0135 - 0.0000i  -0.0026 - 0.0003i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0026 + 0.0003i   0.0017 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

Fields `Dc` and `Dr` should be block-diagonal, hermtian, positive-definite and structured consistently with the structure defined by `blk`. For any matrix `Delta` with the structure defined by `blk`, it should be that `Delta*Dr = Dc*Delta`. The matrices `Dc` and `Dc` should be "essentially" equal, although dimensions will be different if `blk` contains any nonsquare full blocks.

```
VLmi.Dc
```

```
ans =

   0.0063 + 0.0000i  -0.0015 + 0.0003i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0015 - 0.0003i   0.0026 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0018 + 0.0000i  -0.0004 + 0.0001i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0004 - 0.0001i   0.0031 + 0.0000i
```

```
eig(VLmi.Dc)
```

```
ans =

    0.0017
    0.0020
    0.0032
    0.0068
```

```
VLmi.Dr
```

```
ans =

   0.0063 + 0.0000i  -0.0015 + 0.0003i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0015 - 0.0003i   0.0026 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0018 + 0.0000i  -0.0004 + 0.0001i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0004 - 0.0001i   0.0031 + 0.0000i
```

The upper bound is verified by a semidefinite test on a Hermitian matrix formed from the matrix `M`, the upper bound (`bnds(1)`, and the scalings. Construct the matrix, and verify that it is negative semidefinite.

```
T = M'*VLmi.Dr*M - bnds(1)^2*VLmi.Dc + sqrt(-1)*(VLmi.Gcr*M-M'*VLmi.Grc);
eig(T)
```

```
ans =

  -0.0608 - 0.0000i
  -0.0089 + 0.0000i
  -0.0000 - 0.0000i
  -0.0003 - 0.0000i
```

Verfication for only square, complex blocks is easier

Repeat the steps for a problem involving only square complex blocks.

```
blk = [3 0;2 2;1 1];
M = complex(randn(6),randn(6));
[bnds,muinfo] = mussv(M,blk);
[VDelta,VSigma,VLmi] = mussvextract(muinfo);
```

Check perturbation and lower bound

Verify that Delta has correct dimension

```
size(VDelta)
```

```
ans =

     6     6
```

Examine Delta, and confirm it has the proper structure.

```
VDelta
```

```
VDelta =

  Columns 1 through 4

  -0.1873 + 0.0139i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.1873 + 0.0139i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.1873 + 0.0139i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.1517 + 0.0182i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0565 + 0.0242i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 6

   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0783 + 0.0299i   0.0000 + 0.0000i
   0.0212 - 0.0262i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0211 - 0.1867i
```

M*VDelta should have an eigenvalue at 1 (so det(I-M*VDelta)=0)

```
eig(M*VDelta)
```

```
ans =

   1.0000 - 0.0000i
  -0.5452 - 0.1674i
   0.3390 - 0.3155i
   0.0200 + 0.3307i
   0.0000 + 0.0000i
   0.1339 + 0.1010i
```

Lower bound should be equal to 1/norm(VDelta)

```
[bnds(2) 1/norm(VDelta)]
```

```
ans =

    5.3232    5.3232
```

Verify upper bound, using certificates from `VSigma`

Fields `GLeft`, `GMiddle` and `GRight` will all be zero

```
VSigma.GLeft
```

```
ans =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
isequal(VSigma.GLeft,VSigma.GMiddle)
```

```
ans =

  logical

   1
```

```
isequal(VSigma.GRight,VSigma.GMiddle)
```

```
ans =

  logical

   1
```

Fields `DLeft` and `DRight` will be block-diagonal, and structured consistent with the structure defined by `blk`, and equal.

```
VSigma.DLeft
```

```
ans =

  Columns 1 through 4

   0.1048 + 0.0000i  -0.0040 - 0.0019i   0.0173 - 0.0060i   0.0000 + 0.0000i
  -0.0040 + 0.0019i   0.1008 + 0.0000i   0.0006 + 0.0119i   0.0000 + 0.0000i
   0.0173 + 0.0060i   0.0006 - 0.0119i   0.1060 - 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.1025 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 6

   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.1025 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0910 + 0.0000i
```

```
isequal(VSigma.DRight,VSigma.DLeft)
```

```
ans =

  logical

   1
```

With no G-scalings, the upper bound is simply the "DMDinv" norm.

```
[norm(VSigma.DLeft*M/VSigma.DRight) bnds(1)]
```

```
ans =

    5.3232    5.3234
```

Verify upper bound, using certificates from `VLmi`

Fields `Grc`, `Gcr` are zero

```
VLmi.Gcr
```

```
ans =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
isequal(VLmi.Grc,VLmi.Gcr)
```

```
ans =

  logical

   1
```

Fields `Dc` and `Dr` should be block-diagonal, hermitian, positive-definite and structured consistently with the structure defined by `blk`, and equal.

```
VLmi.Dc
```

```
ans =

  Columns 1 through 4

   0.0113 + 0.0000i  -0.0009 - 0.0006i   0.0037 - 0.0013i   0.0000 + 0.0000i
  -0.0009 + 0.0006i   0.0103 + 0.0000i   0.0001 + 0.0025i   0.0000 + 0.0000i
   0.0037 + 0.0013i   0.0001 - 0.0025i   0.0117 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0105 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 6

   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0105 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0083 + 0.0000i
```

```
eig(VLmi.Dc)
```

```
ans =

    0.0070
    0.0083
    0.0099
    0.0105
    0.0105
    0.0164
```

```
isequal(VLmi.Dr,VLmi.Dc)
```

```
ans =

  logical

   1
```

With no G-scalings, the upper bound is simply the "DMDinv" norm. In the LMI certificates, the matrix square roots must be taken of the D matrices.

```
[norm(sqrtm(VLmi.Dr)*M/sqrtm(VLmi.Dc)) bnds(1)]
```

```
ans =

    5.3232    5.3234
```

Upper-Bound quality

By making approximations, the convex optimization for the upper-bound can be solved quickly, at the expense of bound quality. The `'f'` option forces a faster, but less complete minimization, usually resulting in lower quality (ie., larger) upper bounds. Test this option. Run this section several times to see the effect.

```
Delta = [-3 0;-2 0;1 0;1 0];
M = crandn(7,7);
tic; bndDefault = mussv(M,Delta); tDefault = toc;
tic; bndFaster = mussv(M,Delta,'f'); tFaster = toc;
disp(['Default upper bound: ' num2str(bndDefault(1)) ', Time: ' num2str(tDefault)]);
disp(['Faster upper bound: ' num2str(bndFaster(1)) ', Time: ' num2str(tFaster)]);
```

```
Default upper bound: 4.0441, Time: 0.076413
Faster upper bound: 4.6875, Time: 0.019202
```

Ordering of `MU` based on order of `DELTA` set

Create 3 different block structures, each of superset of the preceeding

```
DeltaA = [-1 0;-1 0;1 0];
DeltaB = [-1 0;1 0;1 0];    % contains DeltaA
DeltaC = [-1 0;2 2];        % contains DeltaB
M = crandn(3,3);
% Compute bounds for MU
bndA = mussv(M,DeltaA);
bndB = mussv(M,DeltaB);
bndC = mussv(M,DeltaC);
% Based on Delta, muA <= muB <= muC.  The upper bounds should satisfy this
% ordering as well.
[bndA(1) bndB(1) bndC(1)]
```

```
ans =

    2.0946    2.1133    2.1320
```

Conclusions

The lower and upper bounds for `MUSSV` can be certified using all of the information returned by `mussv` in the 2nd output argument. The lower bound is certified by simply exhibiting a perturbation matrix with the corect structure that causes singularity. The lower bound is then the reciprocal of that matrix. The upper bound is certified using the S-procedure, which amounts to the existence of "D" and "G" matrices with the appropriate structure (consistent with the uncertainty structure) that together with `M`, atisfy an inequality. While the formulae in notes and literature are simple, in practice they are slighly more complicated due to the possibility of non-square full blocks, which make the dimension bookkeeping a bit clumsy.

File Information

```
disp(mfilename)
```

```
verifyMUSSV
```

Lessons learned from the file

In order to use the high-level command mussv, the input argument needs to follow the specific rule stated in the official documentation to create the block structure. The output of the mussv, the muinfo gives us a comprehensive result about the upper bound information (the newlin/young method consisting finding the scailing beta, D, and G, the combination of which is smaller than 1 would certify the upperbound; the semidefinite verification of the upperb ouds is connected to the lecture slides, which seays the G term is included to shift the complex disk to cover the real term in the delta) and lower bound information (VDelta, which is for the lower bound, the product of VDelta and M would make the det(I-M*VDelat = 0), and the system will be unstable, thus we could find the structured delta for the system to be unstable)? If the block is purely complex, then the general formulations of the upperbounds and lowerbounds can be simplified, which is consistent with the lecture notes. Additionally, for three blocks, where one contains another, the largest block would give the smallest minimum singular value, and the reciprocal of which is the uppper bound, and thus the biggest upperbound among the three.

Published with MATLAB® R2017b