SVD decoupling & principal maneuvers: motivation, theory, examples

Objectives: Understand the "intuitive" meaning of principal maneuvres and SVD decoupling, and its relationship with other "common sense" strategies for multivariable control based on:

(*a*) "sensor subset selection" or "weighted sensor averages",

(*b*) "fusing" actuators by "balanced load sharing".

Presentations in video:

https://personales.upv.es/asala/YT/V/dsvdintu1EN.html (theory review)

https://personales.upv.es/asala/YT/V/dsvdintu2EN.html (examples 1 and 3)

https://personales.upv.es/asala/YT/V/dsvdintu3EN.html (example 2)

https://personales.upv.es/asala/YT/V/dsvdintu4EN.html (example 2, realizable reference)

https://personales.upv.es/asala/YT/V/dsvdintu5EN.html (example 4)

https://personales.upv.es/asala/YT/V/dsvdintu6EN.html (manual subset/fusion)

Table of Contents

Theory outline

- Given a "scaled" gain matrix $G_{n\times m}$, its singular value decomposition (svd), that is, $G_{scaled} = U_{n\times n}S_{n\times m}V^T_{m\times m}$ allows the identification of the principal maneuvers (*output directions*: columns of U; *input directions*: columns of V), and their principal *gains*.

- The least squares problems associated with the pseudoinverse of G_{scaled} also have an SVD interpretation, because

$$\text{pinv}(\ G_{scaled}\ ) = V\underline{S}^{-1}U^T,$$

where $\underline{S}^{-1} = pinv(S)$ is calculated by inverting only the non-zero singular values of the diagonal of S (and transposing).

- **SVD decoupling:**

From $G_{scaled} = USV^T$, $y = G_{scaled} \cdot u$, the change of variable:

$$y^{SVD} = U^T y \ , \ u^{SVD} = V^T u$$

allows writing:

$$y^{SVD} = U^T y = U^T G_{esc} u = U^T (USV^T) u = S u^{SVD},$$

thus, the behavior (in static gain) between the "virtual" output y^{SVD} and the "virtual" input u^{SVD} is ***diagonal***, even with ***NON-square*** plants.

Since y is the actually measured thing, $y^{SVD} = U^T y$ must be computed from measurements, so the controllers will calculate:

$$u^{SVD} = K(s) \cdot e^{SVD} = K(s) \cdot U^T \cdot (r - y)$$

with **diagonal** $K(s)$ (hence the name "*decoupling*").

Then the (scaled) physical control action will be $u = V \cdot u^{SVD}$. So, $u^{SVD} = V^T u$ isn't actually used, its "inverse" is used instead.

The SVD-decoupled controller will end up being:

$$u = V \cdot K(s) \cdot e^{SVD} = V \cdot K(s) \cdot U^T \cdot (r - y)$$

If it is decided NOT to control low gain main maneuvers (to avoid saturation/conditioning problems), some elements of the diagonal of $K(s)$ can be forced to zero... or, equivalently, have a smaller-size $K(s)$ and not multiply by the rows and columns associated with UNcontrolled maneuvers, keeping "q" maneuvers under control:

$$u = V_{(:,1:q)} \cdot K_{q \times q}(s) \cdot (U_{(:,1:q)})^T \cdot (r - y)$$

Let us clarify with some simple examples the intuitive meaning of all these procedures.

Example 1: virtual actuator that "combines" two physical actuators (1 x 2 plant)

Let's imagine a process with a resistance that heats, a fan that cools, each with its units:

```
G=[1 -10];
```

Let's do scaling to make things adimensional:

```
Eu=diag([1 0.2]); %scaling to +1/-1.
```

```
Gscaled=G*Eu
```

```
Gscaled = 1×2
     1    -2
```

Fan at "full power" (maximum increment till saturation) has double cooling effect than resistance "at full power" has heating.

```
[U,S,V]=svd(Gscaled)
```

```
U = 1
S = 1×2
    2.2361         0
V = 2×2
    0.4472    0.8944
   -0.8944    0.4472
```

```
norm(Gscaled) %singular values of vector equal to Euclidean norm
```

```
ans = 2.2361
```

```
pinv(Gscaled) %least norm input to achieve unit output increment
```

```
ans = 2×1
    0.2000
   -0.4000
```

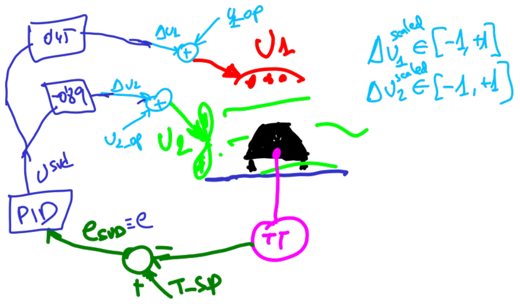
```
V(:,1)/S(1,1) %SVD formula for pseudo-inverse
```

```
ans = 2×1
    0.2000
   -0.4000
```

Open loop: For each unit of output increment (scaled, of course) we desire, we should increase u_1 "20% of full scale increment" and decrement u_2 "40% of full scale increment" .

SVD decoupling (closed loop): It would simply amount to having a PID (SISO) that computed $u^{SVD} = K(s) \cdot (r - y)$, because $y^{SVD} \equiv y$. Then, we would apply it to the actuators with a "load-sharing" strategy:

$$u = \begin{pmatrix} 0.45 \\ -0.89 \end{pmatrix} \cdot u^{SVD}$$


0.45
-0.89
Δu1
Δu2
u1_op
u2_op
+
+
U1
U2
u svd
PID
e_SVD=e
TT
T_SP
+
-
$\Delta u_1^{scaled} \in [-1,+1]$
$\Delta u_2^{scaled} \in [-1,+1]$

The controller $K(s)$ would see a "fictitious" `SISO` plant with gain ***S(1,1)=2.236*** . Indeed:

```
Gscaled*V(:,1)
```

```
ans = 2.2361
```

In principle, roughly speaking, a unit "proportional" gain in $K(s)$ would be what would approximately "saturate" the inputs (formally, $u_1^2+u_2^2=1$) in the face of a unit (scaled) reference increase.

*This strategy may be called "**load balancing**": we distribute the manipulated variable increments proportionally to their effect (after scaling). The SVD or/and the pseudoinverse (trivial, in this case), mathematically formalize the intuitive idea.

We could choose "*split range*" (only act with one MV and connect the second one if the first saturates), or other options (*cascade*) if the dynamics are very different. These issues are out of the SVD-related scope here.

Example 2: plant with 3 controlled outputs x 1 manipulated input

*We'll assume everything scaled unless explicitly discussed.

Assume we have a heating element over a metal piece with three thermocouples whose model is (suitably scaled):

```
G=[1; 2; 1]
```

```
G = 3×1
     1
     2
     1
```

```
norm(G)
```

```
ans = 2.4495
```

```
[U,S,V]=svd(G)
```

```
U = 3×3
    0.4082   -0.8165   -0.4082
    0.8165    0.5266   -0.2367
    0.4082   -0.2367    0.8816
S = 3×1
    2.4495
         0
         0
V = 1
```

```
pinv(G) %least squares fit, we cannot achieve zero error with 3 setpoints
```

```
ans = 1×3
    0.1667    0.3333    0.1667
```

```
1/S(1,1)*U(:,1)' %pseudoinverse via SVD
```

```
ans = 1×3
    0.1667    0.3333    0.1667
```

Interpretation as "projection", SVD decoupling:

The projection of the combination of outputs given by $y^{SVD}_{1\times 1} = (U(:,1)^T)_{1\times 3} \cdot y_{3\times 1}$ is actually the only "controllable" output.

```
U(:,1)'
```

```
ans = 1×3
    0.4082    0.8165    0.4082
```

It is a "weighted average" in with central sensor counts "double".

The regulator* would compute:

$$u = K(s) \cdot e^{SVD} = K(s) \cdot (r^{SVD} - y^{SVD}) = K(s) \cdot U(:,1)^T \cdot (r - y)$$

*In this case $u \equiv u^{SVD}$.

The apparent gain that $K(s)$ will see is $S(1,1)$, because $U^T G = U^T (US) = S$.

```
U(:,1)'*G
```

```
ans = 2.4495
```

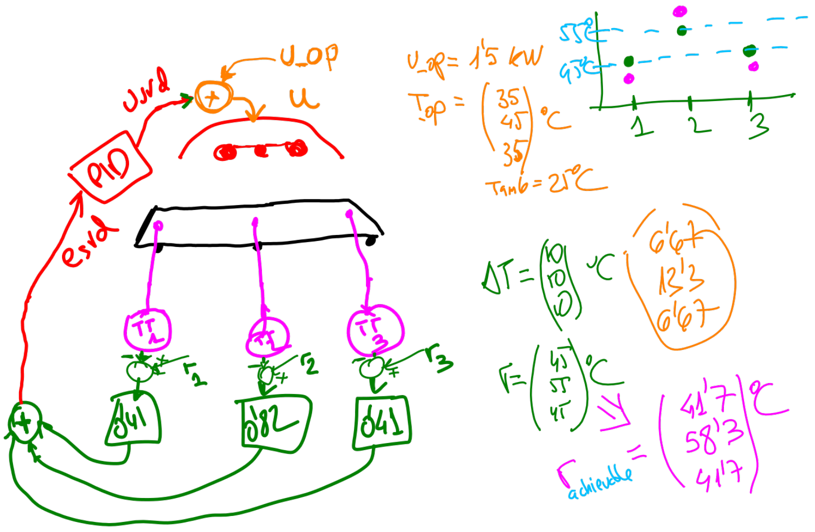
If user inputs three arbitrary references, the only "realizable" feasible reference will be: $r^{feasible} = U(:,1) \cdot y^{SVD} = U(:,1) \cdot U(:,1)^T \cdot r$, as, well, control will achieve "zero" error if: $y^{SVD} = r^{SVD} = U^T_{(:,1)} r$.

The components of r^{SVD} (only one here) are the "length" of the projection of r in the line with direction given by each the chosen columns of U. Multiplying said column of U times such length, we get the actual projection vector in physical units, $y = U_{(:,1)} \cdot y^{SVD} = U_{(:,1)} U^T_{(:,1)} \cdot r$. Matrix $U_{(:,1)} U^T_{(:,1)}$ is a projection matrix.

```
T=U(:,1)*U(:,1)'
```

```
T = 3×3
    0.1667    0.3333    0.1667
    0.3333    0.6667    0.3333
    0.1667    0.3333    0.1667
```

*`T` is the transformation from "physical, user" setpoints to "realizable" ones.



Example:

```
ref=[1;1;1]*10
```

```
ref = 3×1
    10
    10
    10
```

```
U(:,1)'*ref %setpoint to PID under SVD decoupling
```

```
ans = 16.3299
```

```
T*ref % realizable setpoint increment
```

```
ans = 3×1
    6.6667
   13.3333
    6.6667
```

Let us check the relation with least squares (pseudo-inverse):

```
u_opt=pinv(G)*ref
```

```
u_opt = 6.6667
```

```
y_achieved=G*u_opt
```

```
y_achieved = 3×1
    6.6667
   13.3333
    6.6667
```

"Alternate" option to SVD via "common sense", SUBSET:

With a single actuator, choose only one of the three sensors as the controlled variable (the central one, which is the one on which the actuator has the most effect). It would be basically correct... You would have to decide what is more interesting in the specific application, whether to control only the "central temperature" or if you really want to control the "weighted average" of the temperatures (equivalently, reduce error by least squares), so that if any disturbance affects the sides, the control "reacts" to it. The client could want a different "weighting" of the errors... they would change the scaling or, well, in this case, as it is really 1VM, 1VC and all the numbers are >1, the 3 sensors could really be weighted with great freedom.

Example 3: plant with 1 controlled output x 3 manipulated inputs

This is like example 1, but thinking of "three" actuators on a controlled variable, instead of two... but the reasoning is identical.

```
G=[1 2 2]
```

```
G = 1×3
     1     2     2
```

```
norm(G)
```

```
ans = 3
```

```
[U,S,V]=svd(G)
```

```
U = 1
S = 1×3
     3     0     0
V = 3×3
    0.3333   -0.6667   -0.6667
    0.6667    0.6667   -0.3333
    0.6667   -0.3333    0.6667
```

```
pinv(G)
```

```
ans = 3×1
    0.1111
    0.2222
    0.2222
```

```
V(:,1)*1/S(1,1)
```

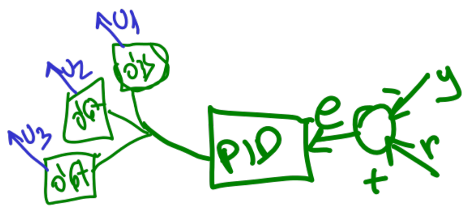
```
ans = 3×1
    0.1111
    0.2222
    0.2222
```

SVD decoupling interpretation: only the actuator maneuver V(:,1) has an effect on the (single) output.

If we wish to increment output in r units, we should act with $pinv(G) \cdot r$ input incremental units; any other input increment achieving r will score a larger $u_1^2 + u_2^2 + u_3^2$.

Or, well, with SVD decoupling,we will set a PID to compute $u^{SVD} = K(s) \cdot (r - y)$, because $y^{SVD} \equiv y$. Then, we would apply it to actual manipulated variables in physical units as:

$$u = \begin{pmatrix} 0.33 \\ 0.67 \\ 0.67 \end{pmatrix} \cdot u^{SVD}$$



If I change variable to $u = V(:,1) \cdot u^{SVD}$ then the apparent gain from u^{SVD} to y will be $S(1,1)$; indeed $y = USV^T u = (US)V^T V(:,1) \cdot u^{SVD} = S_{11} \cdot u^{SVD}$, as $U = 1$. Controller $K(s)$ would see a "fictitious" SISO process with gain ***S(1,1)=3*** .

```
G*V(:,1)
```

```
ans = 3
```

Example 4: full plant, 3 controlled outputs x 4 manipulated inputs

```
G=[1 -4.5 3 -12; 2 -8 2 -8; 3 -12 1 -4.5]
```

```
G = 3×4
    1.0000   -4.5000    3.0000  -12.0000
    2.0000   -8.0000    2.0000   -8.0000
    3.0000  -12.0000    1.0000   -4.5000
```

```
rank(G)
```

```
ans = 3
```

Let us scale to [-1,+1]:

```
Ey=diag([4.5 5 5]);
Eu=diag([1.5 0.6 1.5 0.6]);
Gscaled=inv(Ey)*G*Eu
```

```
Gscaled = 3×4
    0.3333   -0.6000    1.0000   -1.6000
    0.6000   -0.9600    0.6000   -0.9600
    0.9000   -1.4400    0.3000   -0.5400
```

```
[U,S,V]=svd(Gscaled)
```

```
U = 3×3
   -0.6328    0.6815    0.3676
   -0.5466   -0.0569   -0.8354
   -0.5484   -0.7296    0.4085
S = 3×4
    2.9264         0         0         0
         0    1.1381         0         0
         0         0    0.0184         0
V = 4×4
   -0.3528   -0.4074   -0.6028    0.5883
    0.5789    0.6119   -0.3695    0.3922
   -0.3845    0.3764   -0.6036   -0.5883
    0.6265   -0.5638   -0.3685   -0.3922
```

Obviously, apart from a very small minimum gain, the condition number is unacceptable:

```
cond(Gscaled)
```

```
ans = 159.2911
```

If we only wish to control TWO things, that can be done well, according to the condition-number criterion:

```
S(1,1)/S(2,2)
```

```
ans = 2.5712
```

We may carry out SVD decoupling by controlling two principal maneuvres:

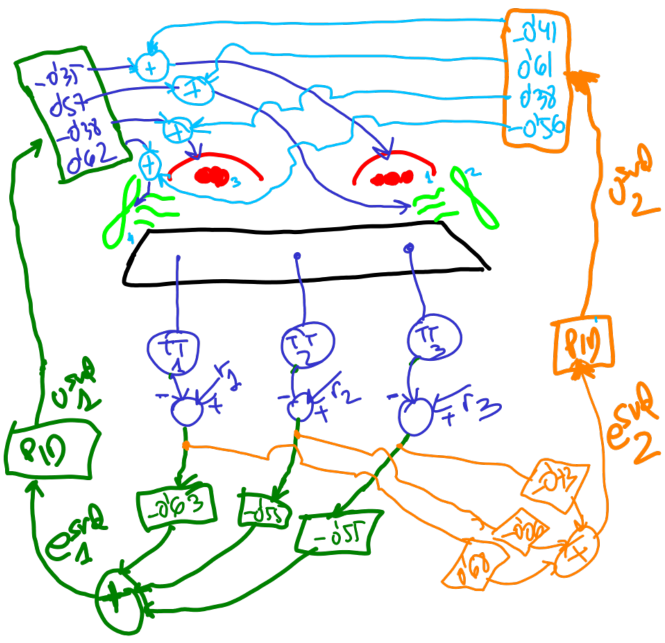
$y^{SVD} = U(:,1:2)^T \cdot y$, and $u^{SVD} = V(:,1:2)^T u$; actually, the second transform will not be used, because we decide u^{SVD} to transform to $u = V(:,1:2) \cdot u^{SVD}$.

```
U(:,[1 2])' %from physical sensors to virtual SVD ones
```

```
ans = 2×3
   -0.6328   -0.5466   -0.5484
    0.6815   -0.0569   -0.7296
```

```
V(:,[1 2]) %from virtual "uSVD" to physical "u".
```

```
ans = 4×2
   -0.3528   -0.4074
    0.5789    0.6119
   -0.3845    0.3764
    0.6265   -0.5638
```



Realizable reference:

```
T=U(:,[1 2])*U(:,[1 2])'
```

```
T = 3×3
    0.8648    0.3071   -0.1502
    0.3071    0.3020    0.3413
```

```
   -0.1502    0.3413    0.8331
```

```
r=[1; .5; 0.];
T*r %realizable
```

```
ans = 3×1
    1.0184
    0.4581
    0.0205
```

Alternative to SVD decoupling (sensor subset): Controlling only left and right sides?.

If we disregard the central "sensor" (at least, as a "controlled variable"; indeed, we may use it on a "state observer" even if it is not a controlled variable, if we have such a measure), we have:

```
subspace(U(:,1:2),[1 0;0 0;0 1])*180/pi %good, angle between coordinate planes and principal maneuvres
```

```
ans = 33.3374
```

```
subspace(U(:,1:2),[0 0;1 0;0 1])*180/pi %not good
```

```
ans = 68.4304
```

```
subspace(U(:,1:2),[1 0;0 1;0 0])*180/pi %not good
```

```
ans = 65.8893
```

```
G2=Gscaled([1 3],:)
```

```
G2 = 2×4
    0.3333   -0.6000    1.0000   -1.6000
    0.9000   -1.4400    0.3000   -0.5400
```

```
[U2,S2,V2]=svd(G2)
```

```
U2 = 2×2
   -0.7623   -0.6472
   -0.6472    0.7623
S2 = 2×4
    2.4510         0         0         0
         0    1.1353         0         0
V2 = 4×4
   -0.3413    0.4143   -0.4447    0.7170
    0.5668   -0.6249   -0.3043    0.4422
   -0.3902   -0.3686    0.7045    0.4642
    0.6402    0.5495    0.4618    0.2737
```

```
cond(G2) %acceptable
```

```
ans = 2.1589
```

If I chose a different sensor selection, I would get a lower "minimum gain":

```
svd(Gscaled([1 2],:))
```

```
ans = 2×1
    2.5100
    0.5423
```

```
svd(Gscaled([2 3],:))
```

```
ans = 2×1
    2.3576
    0.5196
```

which means that I would see "less increments" in my sensors in the worst-case maneuver... the motion I observe with three sensors will be significantly less "observable" with sensors [1 2] or [2 3] than with [1 3].

But we would need four manipulated inputs and SVD decoupling on it, so, well, we could have done it on the original plant $G_{3\times 4}$... maybe selecting only 2 actuators would do?

```
svd(G2(:,[1 2]))
```

```
ans = 2×1
    1.8313
    0.0328
```

```
svd(G2(:,[1 3]))
```

```
ans = 2×1
    1.2707
    0.6296
```

```
svd(G2(:,[1 4]))
```

```
ans = 2×1
    1.8139
    0.6946
```

```
svd(G2(:,[2 3]))
```

```
ans = 2×1
    1.7301
    0.7283
```

```
svd(G2(:,[2 4])) %**** GOOD... almost unit mingain, but not enough
```

```
ans = 2×1
    2.0959
    0.9447
```

```
svd(G2(:,[3 4]))
```

```
ans = 2×1
    1.9851
    0.0302
```

```
G2x2=G2(:,[2 4])
```

```
G2x2 = 2×2
   -0.6000   -1.6000
   -1.4400   -0.5400
```

```
cond(G2x2)
```

```
ans = 2.2185
```

```
RGA=G2x2.*inv(G2x2') %good RGA...
```

```
RGA = 2×2
   -0.1636    1.1636
    1.1636   -0.1636
```

Another alternate option to SVD: Common-sense "actuator fusion"?

```
Gscaled
```

```
Gscaled = 3×4
    0.3333   -0.6000    1.0000   -1.6000
    0.6000   -0.9600    0.6000   -0.9600
    0.9000   -1.4400    0.3000   -0.5400
```

```
V2(:,[1 2])
```

```
ans = 4×2
   -0.3413    0.4143
    0.5668   -0.6249
   -0.3902   -0.3686
    0.6402    0.5495
```

```
V2(2,[1 2])./V2(1,[1 2])
```

```
ans = 1×2
   -1.6608   -1.5083
```

```
V2(4,[1 2])./V2(3,[1 2])
```

```
ans = 1×2
   -1.6406   -1.4907
```

```
F=[1 -1.58 0 0;0 0 1 -1.56]'
```

```
F = 4×2
    1.0000         0
   -1.5800         0
         0    1.0000
         0   -1.5600
```

```
F=F*diag([1/norm([1 -1.58]),1/norm([1 -1.56])]) %por no cambiar "escalado"
```

```
F = 4×2
    0.5348         0
   -0.8450         0
         0    0.5397
         0   -0.8419
```

```
G3=G2*F
```

```
G3 = 2×2
    0.6853    1.8867
    1.6981    0.6165
```

G3 is quite similar to the previous G2x2 because one actuator "dominates" a lot in the weighting... If it is a prototype, a plant could be designed with only 2 actuators... But if I already have the 4 of them, we move on.

```
svd(G3)
```

```
ans = 2×1
    2.4504
    1.1350
```

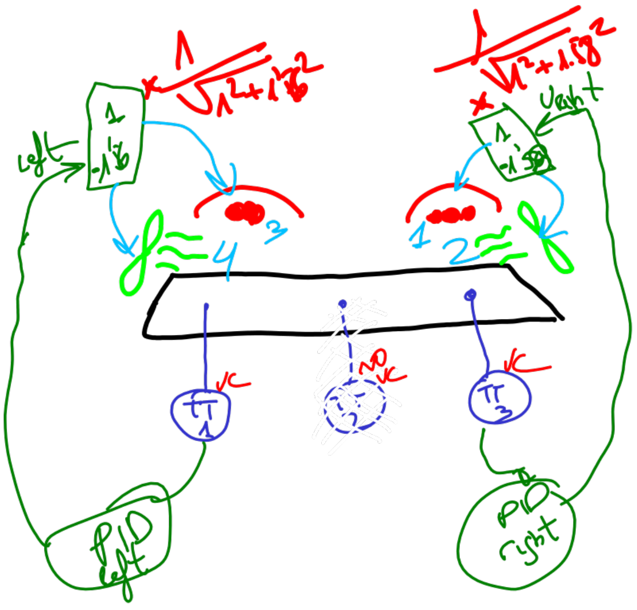
```
cond(G3)
```

```
ans = 2.1589
```

```
RGA=G3.*inv(G3')
```

```
RGA = 2×2
   -0.1519    1.1519
    1.1519   -0.1519
```

RGA is reasonable, we could try PIDs with those "fused" actuators two by two... or the "fusion of 4 actuators" given by the SVD of G2, of course... OR/AND the fusion of 3 sensors given by the SVD by original Gesc, obviously.



$\times \frac{1}{\sqrt{1^2+1.56^2}}$
$\times \frac{1}{\sqrt{1^2+1.58^2}}$
left
right
1
-1.56
1
-1.58
3
4
1
2
π_1
π_2
π_3
✓
NO ✓
✓
PID left
PID right