Supplement to

"*Positive linear functionals without representing measures*"

Eq. 2.6 in the paper defines ψ as shown below. **Proposition 2.2** in the paper states that ψ is independent of the variables *s* and *t*. The computational proof of this result using Maxima is explained in detail below. The actual Maxima commands are shown in blue color.

We start from the matrices J and W:

$$J = \begin{pmatrix} 1 & a & b & c & e & d & f & gg & x \\ a & c & e & b & f & gg & d & h & j \\ b & e & d & f & gg & x & h & j & k \\ c & b & f & e & d & h & gg & x & u \\ e & f & gg & d & h & j & x & u & v \\ d & gg & x & h & j & k & u & v & w \\ f & d & h & gg & x & u & j & k & r \\ gg & h & j & x & u & v & k & r & s \\ x & j & k & u & v & w & r & s & t \end{pmatrix}$$

and

$$W = \begin{pmatrix} h \\ x \\ u \\ j \\ k \\ r \\ v \\ w \end{pmatrix}$$

Following the notation in the paper, we write

$$J^{-1} = \begin{pmatrix} P & V \\ V^T & \epsilon \end{pmatrix} = \frac{1}{D} \begin{pmatrix} P' & V' \\ V'^T & \epsilon' \end{pmatrix}.$$

Here $\begin{pmatrix} V \\ \epsilon \end{pmatrix}$ form the last column of J^{-1}, and D is the determinant of J that we factor out from J^{-1}. Then,

$$\psi = \frac{\langle P'W,W\rangle - \frac{1}{\epsilon'}\langle V'^T,W\rangle^2}{D} \ \ldots \ \text{Eq (2.6) from the paper}$$

Proposition 2.2 in the paper claims that ψ is independent of s and t.

Let

$$\langle P'W, W\rangle = P_1 = \alpha_0 + \alpha_1 s + \alpha_2 s^2 + \alpha_3 t$$

$$\langle V^{T\prime}, W\rangle = P_2 = \beta_0 + \beta_1 s$$

$$N = Numerator(\psi) = P_1 - \frac{1}{\epsilon'} P_2^2$$

$$N = (\alpha_0 - \frac{1}{\epsilon'}\beta_0^2) + (\alpha_1 - \frac{2}{\epsilon'}\beta_0\beta_1)\ s + (\alpha_2 - \frac{1}{\epsilon'}\beta_1^2)\ s^2 + \alpha_3\ t$$

$$D = Denominator(\psi) = \gamma_0 + \gamma_1\ s + \gamma_2\ s^2 + \gamma_3\ t$$

N and D above will enable us to form Eq. (2.7) in the paper:

$$\psi = \frac{N}{D} = \frac{n_0 + n_1 s + n_2 s^2 + n_3 t}{d_0 + d_1 s + d_2 s^2 + d_3 t} \quad \ldots \quad \text{Eq. (2.7) from the paper}$$

where:

$n_0 = (\alpha_0 - \frac{1}{\epsilon'}\beta_0^2)$; $n_1 = (\alpha_1 - \frac{2}{\epsilon'}\beta_0\beta_1)$; $n_2 = (\alpha_2 - \frac{1}{\epsilon'}\beta_1^2)$;
$n_3 = \alpha_3$; $d_0 = \gamma_0$;$d_1 = \gamma_1$; $d_2 = \gamma_2$; $d_3 = \gamma_3$

Then:

$$f_0 = \frac{n_0}{d_0}\ , f_1 = \frac{n_1}{d_1}\ , f_2 = \frac{n_2}{d_2}\ , f_3 = \frac{n_3}{d_3}$$

and our goal is to show that f_0, f_1, f_2, f_3 are all equal by showing that

$$\frac{f_0}{f_3} = \frac{f_1}{f_3} = \frac{f_2}{f_3} = 1 \qquad \ldots \qquad \text{Eq. (2.9) from the paper}$$

For easy reference we repeat the representations of J^{-1} and ψ:

$$J^{-1} = \begin{pmatrix} P & V \\ V^T & \epsilon \end{pmatrix} = \frac{1}{D} \begin{pmatrix} P' & V' \\ V'^T & \epsilon' \end{pmatrix}$$

$$\psi = \frac{\langle P'W,W\rangle - \frac{1}{\epsilon'}\langle V'^T,W\rangle^2}{D} \ \ldots \ \text{Eq (2.6) from the paper}$$

The table below shows the correspondence between the symbols used in the paper and the Maxima variables we use:

Symbol from paper	**Variable name used in Maxima program**
J^{-1}	inv
D	D
P'	rho_p
$P'W$	rhopW
$\langle P'W, W\rangle$	rhopWW
ϵ'	epsilon_p
V'^T	V_p
$\langle V'^T, W\rangle$	VpW

/** **Beginning of Maxima commands. All Maxima commands are in blue. **/

/** Turn on timing of calculations - each calculation is then timed for cpu time and elapsed time. **/

```
showtime:true$
```

/** define matrices J and W **/

```
J: matrix([1, a, b, c, e, d, f, gg, x], [a, c, e, b, f, gg, d, h, j], [b, e, d, f, gg, x, h, j, k], [c, b, f, e, d, h, gg, x, u], [e, f, gg, d, h, j, x, u, v], [d, gg, x, h, j, k, u, v, w], [f, d, h, gg, x, u, j, k, r], [gg, h, j, x, u, v, k, r, s], [x, j, k, u, v, w, r, s, t] );
W:transpose([h,x,u,j,k,r,v,w])$
```

/** The next 3 commands allow factoring out the determinant of J from J^{-1}.

detout environment variable allows the determinant to be factored out from the inverse, and it needs the doallmxops and doscmxops to be set to false.

**/

```
doallmxops: false$
doscmxops: false$
detout: true$
```

/** numInv is the Adjoint of J, and D is the determinant of J. ***/

```
inv:invert(J)$
numInv:num(inv)$
D:denom(inv)$
```

/** reset the matrix computation parameters **/

```
doallmxops: true$
doscmxops: true$
```

/** compute the components that go in to ψ , equation (2.7) **/

```
rho_p:submatrix(9,numInv,9)$
rhopW:rho_p.W$
rhopWW:rhopW.W$
epsilon_p:numInv[9,9]$
```

V_p:submatrix(9,col(numInv,9))$

VpW:transpose(V_p).W$

/** isolate s, s^2, t coefficients as well as terms independent of s and t in rhopWW .

$$\langle P'W, W\rangle = P_1 = \alpha_0 + \alpha_1 s + \alpha_2 s^2 + \alpha_3 t$$

$$\alpha_0 = \mathrm{rhopWW\_const}$$
$$\alpha_1 = \mathrm{rhopWW\_Cs1}$$
$$\alpha_2 = \mathrm{rhopWW\_Cs2}$$
$$\alpha_3 = \mathrm{rhopWW\_Ct1}$$

**/

rhopWW_Cs1:ratcoef(rhopWW, s, 1)$

rhopWW_Cs2:ratcoef(rhopWW, s, 2)$

rhopWW_Ct1:ratcoef(rhopWW, t, 1)$

rhopWW_const:expand(rhopWW-rhopWW_Cs1*s-rhopWW_Cs2*s^2 rhopWW_Ct1 * t)$

/** isolate s, s^2, t coefficients as well as terms independent of s and t in D:

$$D = \gamma_0 + \gamma_1\ s + \gamma_2\ s^2 + \gamma_3\ t$$
$$\gamma_0 = \mathrm{D\_const}$$
$$\gamma_1 = \mathrm{D\_Cs1}$$
$$\gamma_2 = \mathrm{D\_Cs2}$$
$$\gamma_3 = \mathrm{D\_Ct1}$$

**/

D_Ct1:ratcoef(D,t,1)$

D_Cs1:ratcoef(D,s,1)$

```
D_Cs2:ratcoef(D,s,2)$
D_const:expand(D-D_Cs1*s-D_Cs2*s*s-D_Ct1*t)$
```

/** VpW has an s^1 term, and terms independent of s and t. Isolate these.

$$\beta_0 = \mathrm{VpW\_const}$$
$$\beta_1 = \mathrm{VpW\_Cs1}$$

**/

```
VpW_Cs1:ratcoef(VpW,s,1)$
VpW_const:expand(VpW-VpW_Cs1*s)$
```

/** epsilon_p is independent of s and t. We redefine epsilon_p as ep for convenience **/

```
ep:epsilon_p$
```

/** compose the terms f0, f1, f2, f3 for equation (2.9)

$$f_0 = \frac{n_0}{d_0} = \frac{(\alpha_0 - \frac{1}{\epsilon'}\beta_0^2)}{\gamma_0}$$
$$f_1 = \frac{n_1}{d_1} = \frac{(\alpha_1 - \frac{2}{\epsilon'}\beta_0\beta_1)}{\gamma_1}$$
$$f_2 = \frac{n_2}{d_2} = \frac{(\alpha_2 - \frac{1}{\epsilon'}\beta_1^2)}{\gamma_2}$$
$$f_3 = \frac{n_3}{d_3} = \frac{\alpha_3}{\gamma_3}$$

**/

```
f0:(rhopWW_const-(1/ep)*(VpW_const)^2)/D_const$
f1:(rhopWW_Cs1-(2/ep)*(VpW_const*VpW_Cs1))/D_Cs1$
f2:(rhopWW_Cs2-(1/ep)*(VpW_Cs1*VpW_Cs1))/D_Cs2$
f3:rhopWW_Ct1/D_Ct1$
```

/** you can store these values in their non-simplified form to disk so that you do not have to recalculate them again. This is particularly true for f0, which is time and space intensive to simplify. To save f0 to disk, for example, do the next command.

Note: "save" saves the expression in maxima lisp format, which you can readily load back into maxima with the "loadfile" command. However, if you want a readable text output, you need to use the "stringout" command on the expression.

**/

save("/home/easwaran/L/2009/f0.maxima", f0)$

/**Simplify f1, f2, f3.

Warning: Trying to simplify f0 will hang unless you do as described below. **/

ratsimped_f1:ratsimp(f1)$

ratsimped_f2:ratsimp(f2)$

ratsimped_f3:ratsimp(f3)$

/** save these simplified expressions **/

save("/home/easwaran/L/2009/ratsimpedf1.maxima", ratsimped_f1)$

save("/home/easwaran/L/2009/ratsimpedf2.maxima", ratsimped_f2)$

save("/home/easwaran/L/2009/ratsimpedf3.maxima", ratsimped_f3)$

/** The next step is the most time-consuming one. It will not work unless your Lisp Engine is allowed to use large temporary heap space, and your maxima temporary directory has large enough storage. Here is how we proceeded:

1. Recompile SBCL, as explained in the SBCL web page

http://sbcl.sourceforge.net/getting.html under the heading "Installing from source". When you configure your make environment, you can specify

"--dynamic-space-size 25600" to allow the Lisp engine to use up to 25GB of temporary heap space. Install the re-compiled SBCL following instructions in the same web URL mentioned above, under the section "Installing from Binary".

2. By default Maxima uses your home directory (on Unix systems) as its temporary working directory. You can re-set it by the Maxima command:

maxima_tempdir="/path/to/new/tempdir"

where the quoted path is a directory on a file system with large enough space, and where the user has read/write permissions.

After recompiling SBCL and installing it, start Maxima again, adjust the *maxima_tempdir* if necessary, and then read from disk the stored value of f0, and simplify it.

**/

```
loadfile("/home/easwaran/L/2009/f0.maxima")$
ratsimped_f0:ratsimp(f0)$
```

/** After f0 is simplified, you can load back simplified versions of f1, f2 and f3, and check f0/f3, f1/f3, f2/f3. These will all be equal to 1 as claimed in the paper **/

```
loadfile("/home/easwaran/L/2009/f1.maxima")$
loadfile("/home/easwaran/L/2009/f2.maxima")$
loadfile("/home/easwaran/L/2009/f3.maxima")$
```

/** These are all 1, completing the proof of Proposition 2.2 **/

```
ratsimp(f0/f3);
ratsimp(f1/f3);
ratsimp(f2/f3);
```

Here is a link to the actual expression f0=f1=f2=f3.

Here is a sample maxima session output.

PDF version of this file.

HTML version of this file.