==========

Structure factor for a ternary electrolyte mixture

```
> restart:
> with(LinearAlgebra):
> m:=Vector([1,2,3]); # Molecular masses
```

$$m := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{1}$$

```
> w:=Vector([0.2, 0.35, 0.45]); # Mass fractions
```

$$w := \begin{bmatrix} 0.2 \\ 0.35 \\ 0.45 \end{bmatrix} \tag{2}$$

```
> Rho:=3; # Total density
```

$$P := 3 \tag{3}$$

```
> rho:=Rho*w; # Partial densities
```

$$\rho := \begin{bmatrix} 0.600000000000000 \\ 1.05000000000000 \\ 1.35000000000000 \end{bmatrix} \tag{4}$$

```
> rhobar:=Vector([2.0,3.0,3.85714]); # For lowMach
```

$$rhobar := \begin{bmatrix} 2.0 \\ 3.0 \\ 3.85714 \end{bmatrix} \tag{5}$$

For EOS we need inverses of rho_bar's:

```
> rhobar_inv:=map(x->1/x, rhobar);
```

$$rhobar\_inv := \begin{bmatrix} 0.5000000000 \\ 0.3333333333 \\ 0.2592594513 \end{bmatrix} \tag{6}$$

Charged species test case

```
> z_:=Vector(3,[2,1,z3]):
> z3:=solve(Transpose(z_).w=0, z3);
```

$$z3 := -1.666666667 \tag{7}$$

```
> z_; # Charges per mass
```

$$\begin{bmatrix} 2 \\ 1 \\ -1.666666667 \end{bmatrix} \tag{8}$$

For ideal mixtures we have an explicit form for structure factor

1. No charges ideal mixture:

```
> S_w_id:=(IdentityMatrix(3)-w.Transpose(Vector(3,1))).
  DiagonalMatrix(Vector(3,i->w[i]*m[i])).Transpose(IdentityMatrix
  (3)-w.Transpose(Vector(3,1)))/Rho;
```

$$S\_w\_id := \begin{bmatrix} 0.0700000000000000 & -0.0175000000000000 & -0.0525000000000000 \\ -0.0175000000000000 & 0.161875000000000 & -0.144375000000000 \\ -0.0525000000000000 & -0.144375000000000 & 0.196875000000000 \end{bmatrix} \quad (9)$$

And structure factor for total density rho:

```
> S_rho_z_id:=Rho^4*Transpose(rhobar_inv).S_w_id.rhobar_inv;
```

$$S\_rho\_z\_id := 0.349999274044588082 \quad (10)$$

2. **No charges non-ideal mixture**

This was computed earlier for 2015 paper (see Amit.mw)

```
> S_w := Matrix(3, 3, {(1, 1) = 0.5541976620e-1, (1, 2) =
  -0.2417640633e-3, (1, 3) = -0.5517800210e-1, (2, 1) =
  -0.2417640633e-3, (2, 2) = 0.9323259833e-1, (2, 3) =
  -0.9299083420e-1, (3, 1) = -0.5517800210e-1, (3, 2) =
  -0.9299083420e-1, (3, 3) = .1481688363}); # Static structure
  factors non-ideal case
```

$$S\_w := \begin{bmatrix} 0.05541976620 & -0.0002417640633 & -0.05517800210 \\ -0.0002417640633 & 0.09323259833 & -0.09299083420 \\ -0.05517800210 & -0.09299083420 & 0.1481688363 \end{bmatrix} \quad (11)$$

```
> S_rho:=Rho^4*Transpose(rhobar_inv).S_w.rhobar_inv;
```

$$S\_rho := 0.300902667950801506 \quad (12)$$

2. With charges ideal mixture:

```
> S_w_z_id:=S_w_id - S_w_id.z_.Transpose(z_).S_w_id / (Transpose
  (z_).S_w_id.z_); # In the presence of charges
```

$$S\_w\_z\_id := \begin{bmatrix} 0.0448000000013440 & -0.0615999999997480 & 0.0167999999984040 \\ -0.0615999999997480 & 0.0846999999967660 & -0.0230999999970180 \\ 0.0167999999984040 & -0.0230999999970180 & 0.00629999999861403 \end{bmatrix} \quad (13)$$

```
> Transpose(z_).S_w_z_id.z_; # Confirm charge neutral
```

$$1.68268177185363\ 10^{-16} \quad (14)$$

Structure factor for total density rho:

```
> S_rho_z_id:=Rho^4*Transpose(rhobar_inv).S_w_z_id.rhobar_inv;
```

$$S\_rho\_z\_id := 0.0700000726432194603 \quad (15)$$

3. With charges non-ideal mixture

```
> S_w_z:=S_w - S_w.z_.Transpose(z_).S_w / (Transpose(z_).S_w.z_); #
  In the presence of charges
```

(16)

$$S\_w\_z := \begin{bmatrix} 0.0261818181798331 & -0.0359999999900849 & 0.00981818182672957 \\ -0.0359999999878882 & 0.0495000000082877 & -0.0134999999784314 \\ 0.00981818181588683 & -0.0134999999868089 & 0.00368181821587624 \end{bmatrix} \quad \textbf{(16)}$$

```
> Transpose(z_).S_w_z.z_; # Confirm charge neutral
```

$$6.02315760988784\ 10^{-11} \quad \textbf{(17)}$$

Structure factor for total density rho:

```
> S_rho_z:=Rho^4*Transpose(rhobar_inv).S_w_z.rhobar_inv;
```

$$S\_rho\_z := 0.0409091341858012422 \quad \textbf{(18)}$$

Testing stuff--UNFINISHED

```
==================================
> #P1:=(IdentityMatrix(3)-w.Transpose(Vector(3,1)));
> #P2:=(IdentityMatrix(3)-S_w_id.z_.Transpose(z_)/(Transpose(z_).
  S_w_id.z_));
> #P1.P2-P2.P1; # Confirm these commute
> #S_w_z:=P2.P1.DiagonalMatrix(Vector(3,i->w[i]*m[i]/Rho)).
  Transpose(P1).Transpose(P2);
```