Sometimes it is useful to scale a transfer function matrix with a scaling matrix:

$$\hat{\underline{\underline{G}}}(j\omega) = \underline{\underline{D}}(j\omega) \cdot \underline{\underline{G}}(j\omega) \cdot \underline{\underline{D}}^{-1}(j\omega)$$

D is usually chosen as a diagonal constant matrix.

$$\Rightarrow \quad \|\underline{\underline{G}}\|_{\mu} \stackrel{!}{=} \inf_{D}\left(\|\hat{\underline{\underline{G}}}(j\omega)\|_{\infty}\right)$$

This works only if the system has the same number of inputs as outputs, i.e., G(s) is a square rational function matrix.

The above norm is used in the so-called <u>Mu-synthesis</u> design.

It is sometimes useful to also define norms of polynomial matrices, although these have no physical significance. This is useful for numerical purposes. E.g. we may want to define:

$$\mathrm{tol} = \|P\| \cdot \mathrm{sqrt}(\varepsilon)$$

for numerical algorithms operating on P. We define:

$$\|P\|_p \stackrel{!}{=} \left\| \left(\left[\|p_{ij}\|_p \right] \right) \right\|_p$$

The p-norm of a polynomial matrix is defined as the p-norm of the matrix constructed from the p-norms of the individual polynomials.

Given:

p in mode 1:

$V(p)$ = vector of coefficients

Example: $p = s^2 + 7s + 3$

$\Rightarrow V(p) = [\,3 \quad 7 \quad 1\,]$

p in mode 2:

$V(p)$ = vector consisting of gain value and roots with multiplicities

Example: $p = 17(s+1)^2(s+5)$

$\Rightarrow V(p) = [\,17 \quad -1 \quad -1 \quad -5\,]$

p in mode 3:

$V(p)$ = vector of polynomial values evaluated for the domain values.

$$\Rightarrow \|P_{ij}\|_p \overset{!}{=} \|V(P_{ij})\|_p$$

In Polpac:

```
function [n] = nnorm(p,tp)
%
%  Find the norm of a POLPAC data structure.
%
%  Input Parameters:
%  -----------------
%
%    p     :=  Any POLPAC data structure (all modes)
%    tp    :=  The type of mode to be computed.
%              default:  tp = 2
%
%  Output Parameter:
%  -----------------
%
%    n     :=  The norm of p.
%
%  Explanations:
%  -------------
%
%  P can be either a signal (mode = 3, real-valued domain), or a system
%  modes 0, 4, 5, 6, 7, 8, 9.  If P is in modes 1, 2, or 3, and isn't a
%  signal, apply polynomial matrix norms.
%
%  If P is a signal, the following options are available:
%
%    tp = 1     :  Use l1-norm:  n = sum(abs(pi))        over the stored values
%    tp = 2     :  Use l2-norm:  n = sqrt(sum(abs(pi)^2))              "
%    tp = p     :  Use lp-norm:  n = (sum(abs(pi)^p))^(1/p))           "
%    tp = inf   :  Use loo-norm: n = max(abs(pi))                      "
%    tp = 'inf' :  Same as above
%    tp = 'pow' :  nnorm(P,2)/(tmax - tmin)
%    tp = 'spc' :  Calculate the spectrum.
%                  Then compute: n = sqrt(||Suu||oo)
%
%  If P is a signal matrix:
%
%    tp = 2     :  Use l2-norm:  n = sqrt(trace(pi'*pi)) over the stored values
%    tp = inf   :  Use loo-norm: n = max(max(svd(p)))                  "
%    tp = 'inf' :  For each domain value, t0, the matrix of p(t0) is extracted.
%                  max(svd(p)) is the largest singular value of p(t0).
%                  The outer max operator is then taken over all time values.
%    tp = 'pow' :  norm( [ nnorm(pij,'pow') ] , 1)
%    tp = 'spc' :  norm( [ nnorm(pij,'inf') ] , inf)
```

```
%
% If P is a stable system, all system modes except mode = 6 with strictly
% imaginary domain:
%
%   tp = 2       :  Use L2-norm:
%                   Convert system to state-space description.
%                   Get minimal realization.
%                   Compute the controllability Gramian, Gc.
%                   Then compute: n = sqrt(trace(C*Gc*C')
%   tp = inf     :  Use Hoo-norm:
%   tp = 'inf'   :  Solve a series of Riccati equations, until H no longer
%                   has eigenvalues on the imaginary axis.
%   tp = 'hnk'   :  Use Hankel-norm:
%                   n = sqrt(max(eig(Gc*Go)))
%
```

```
% If P is a stable system with mode = 6 and a strictly imaginary domain:
%
%   tp = inf     :  Use Hoo-norm:
%   tp = 'inf'   :  Calculate series of matrices for each domain value.
%                   Then compute: n = max(max(svd(abs(p(jw)))))
%                   where max(svd(.)) is the largest singular value, and the
%                   outer max operator is taken over all stored frequency values.
%   tp = 'mu'    :  Use mu-norm:
%                   n = inf(||D*P/D||oo)        over all nonsingular D matrices
%
% If P is a polynomial matrix:
%
%   There is no physical interpretation of the norm of a polynomial.
%   However, it is useful to define a meaning, because this can be used to
%   assess the "magnitude" of the data values.  For this reason, the norm
%   of a polynomial is defined as follows:
%
%     mode = 1 :  Norm of vector of coefficients.
%     mode = 2 :  Norm of vector consisting of [ gain , l1 (m1 times) ,
%                 l2 (m2 times) , ... ].
%     mode = 3 :  Norm of vector of polynomial values at given domain values.
%
%   The absolute value of a polynomial matrix is the matrix of the absolute
%   values of the individual polynomials.
%
%   tp = 1       :  Use 1-vector/matrix norm.
%   tp = 2       :  Use 2-vector/matrix norm.
%   tp = p       :  Use p-vector/matrix norm.
%   tp = inf     :  Use oo-vector/matrix norm.
%   tp = 'inf'   :  Same as above.
%
```

Computation:

(1) $\mathcal{H}_2$-Norm:

$$\underline{\underline{G}}(t) = C e^{At} B$$

$$\Rightarrow \|\underline{\underline{G}}\|_2 = \sqrt{\int_0^\infty \text{Trace}\{\underline{\underline{G}}^*(t) \cdot \underline{\underline{G}}(t)\} dt}$$

$$= \sqrt{\int_0^\infty \text{Trace}\{B^* e^{A^* t} C^* C e^{At} B\} dt}$$

$$= \sqrt{\text{Trace}\{B^* \int_0^\infty e^{A^* t} C^* C e^{At} dt \cdot B\}}$$

$$= \sqrt{\text{Trace}(B^* G_o \cdot B)}$$

$$= \sqrt{\int_0^\infty \text{Trace}\{\underline{\underline{G}}(t) \cdot \underline{\underline{G}}^*(t)\} dt}$$

$$= \sqrt{\int_0^\infty \text{Trace}\{C e^{At} B B^* e^{A^* t} C^* dt}$$

$$= \sqrt{\text{Trace}\{C \cdot \int_0^\infty e^{At} B B^* e^{A^* t} dt \cdot C^*\}}$$

$$= \sqrt{\text{Trace}(C \cdot G_c \cdot C^*)}$$

(2) $\mathcal{H}_\infty$-Norm:

This cannot be computed in closed form. We need to find a range first. It can be shown that:

$$\|G(j\omega)\|_\infty \geq \max\{\overline{\sigma}_H(S), \overline{\sigma}(D)\}$$

$$\|G(j\omega)\|_\infty \leq \overline{\sigma}(D) + 2 \cdot \sum_{i=1}^{n} \overline{\sigma}_H(S)$$

We can use a series of generalized Riccati equations to find $\|G(j\omega)\|_\infty$.

Choose: $\gamma = 0.5\left(\|G(j\omega)\|_{\infty_{min}} + \|G(j\omega)\|_{\infty_{max}}\right)$

Then: $R = \gamma^2 I - D^* D$.

$$H = \begin{bmatrix} (A + BR^{-1}D^*C) & (BR^{-1}B^*) \\ (-C^*(I + DR^{-1}D^*)C) & (-(A + BR^{-1}D^*C)^*) \end{bmatrix}$$

If eig(H) has no solutions on the imaginary axis:

$\Rightarrow$ γ is too big $\Rightarrow G_{max} = \gamma$

else: γ is too small $\Rightarrow G_{min} = \gamma$

Repeat until $G_{max} - G_{min}$ is sufficiently small.

```
if strcmp(tp,'inf'),
  %
  %  Get Hankel singular values to estimate gamma
  %
  Gc = gram(a,b);
  Go = gram(a',c');
  sH = sqrt(eig(Gc*Go));
  s = svd(d);
  gmin = max([sH(1),s(1)]);
  gmax = s(1) + 2*sum(sH);
  while gmax - gmin > 1.0e-6,
    gamma = 0.5*(gmin + gmax);
    dd = d'*d;
    [nn,m] = size(dd);
    R = gamma*gamma*eye(m) - dd;
    br = b/R;
    h11 = a + br*d'*c;
    h12 = br*b';
    drd = d/R*d';
    [nn,m] = size(drd);
    h21 = -c'*(eye(m) + drd)*c;
    h22 = -h11';
    H = [ h11 , h12 ; h21 , h22 ];
    l = eig(H);
    l = abs(real(l));
    if min(l) > 1000*eps,
      gmax = gamma;
      else
      gmin = gamma;
    end,
  end,
  n = gamma;
  pull,
  return,
end,
```

Of course, if we are in mode=6, and we have a strictly imaginary domain, we can simple find $\bar{\sigma}[G(j\omega)]$ for each domain value and take the maximum of those.

```
if strcmp(tp,'inf'),
  if mod == 6,
    if abs(real(dom)) < 1000*eps,
      pp = zeros(1:logcol);
      if lrow*lcol == 1,
        for i=1:logcol,
          pp(i) = abs(p(3,i))/abs(p(4,i));
        end,
        else
        for i=1:logcol,
          a = p(3:lrow+2,i:logcol:(lcol-1)*logcol+i);
          b = p(lrow+3:2*lrow+2,i:logcol:(lcol-1)*logcol+i);
          g = a ./ b;
          s = svd(g);
          pp(i) = s(1);
        end,
      end,
      n = max(pp);
      pull,
      return,
    end,
  end,
end,
```

(3) μ-Norm:

This norm is currently only implemented for the case of mode = 6 with imaginary domain.

```
if strcmp(tp,'mu'),
  if mod == 6,
    if abs(real(dom)) < 1000*eps,
      pp = zeros(1:logcol);
      if lrow*lcol == 1,
        for i=1:logcol,
          pp(i) = abs(p(3,i))/abs(p(4,i));
        end,
        else
        for i=1:logcol,
          a = p(3:lrow+2,i:logcol:(lcol-1)*logcol+i);
          b = p(lrow+3:2*lrow+2,i:logcol:(lcol-1)*logcol+i);
          g = a ./ b;
          pp(i) = munorm(g);
        end,
      end,
      n = max(pp);
      pull,
      return,
    end,
  end,
end,
```

It is computed in the same way as the infinity norm (in fact, it <u>is</u> an infinity norm). The question is how to approximate the optimal scaling matrix D. We only consider the case of a constant diagonal scaling matrix.

Algorithm.

Start with $D = I^{(n)}$.

Let: $D(i,i) = d_i$

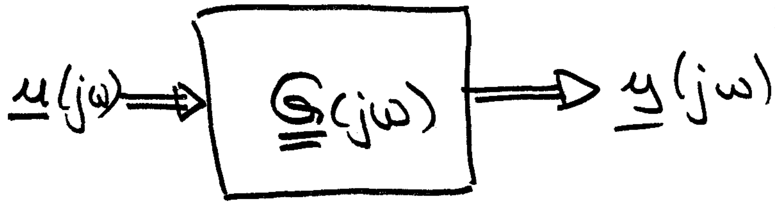
$$\Rightarrow \quad \hat{d}_k^4 = \frac{\sum_{i=1, i\neq k}^{n} \|g_{ik}\|^2 \cdot d_i^2}{\sum_{j=1, j\neq k}^{n} \|g_{kj}\|^2 / d_j^2} \; ; \; k = 1, 2, \ldots, n.$$

Then: $d_i \equiv \hat{d}_i$, and repeat until convergence.

(p. 296 of Zhou/Doyle/Glover).

```
function [mu] = munorm(a)
%
%  Compute the mu-norm of a square matrix A.  The mu-norm is the largest
%  singular value of D*A/D, where D is any suitable scaling matrix.
%
%  Input Parameter:
%  ----------------
%
%    A      :=  square Matlab matrix
%
%  Output Parameter:
%  -----------------
%
%    muval  :=  mu-norm of A
%
push('MUNORM')
global debg
%
[n,m] = size(a);
if debg == 3,
  if n ~= m,
    disp('MUNORM: Error - Mu-norm only defined for square matrices'),
    abort,
    return,
  end,
end
%
%  Compute the optimal scaling matrix
%
d2 = ones(n,1);
d2new = ones(n,1);
aa = abs(a);
aa = aa .* aa;
er = 1000;
while er > 1.0e-6,
  ai = aa(2:n,1) .* d2(2:n);
  aj = aa(1,2:n) ./ d2(2:n)';
  d2new(1) = sqrt(sum(ai)/sum(aj));
  for k=2:n-1,
    ai = aa([1:k-1,k+1:n],k) .* d2([1:k-1,k+1:n]);
    aj = aa(k,[1:k-1,k+1:n]) ./ d2([1:k-1,k+1:n])';
    d2new(k) = sqrt(sum(ai)/sum(aj));
  end,
  er = norm(d2new - d2);
  d2 = d2new;
end
d = sqrt(d2);
d = diag(d);
aa = d*a/d;
s = svd(aa);
mu = s(1);
pull
return
```

Usefulness of systems and signal norms for estimating magnitude of behavior



$$\underline{y}(j\omega) = \underline{\underline{G}}(j\omega) \cdot \underline{u}(j\omega)$$

$$\Rightarrow \|\underline{y}(j\omega)\|_2 = \|\underline{\underline{G}}(j\omega) \cdot \underline{u}(j\omega)\|_2$$

$$= \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \underline{u}(j\omega)^* \cdot \underline{\underline{G}}^*(j\omega) \cdot \underline{\underline{G}}(j\omega) \cdot \underline{u}(j\omega)\, d\omega}$$

$$\leq \|G(j\omega)\|_\infty \cdot \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \underline{u}(j\omega)^* \underline{u}(j\omega)\, d\omega}$$

$$= \|G(j\omega)\|_\infty \cdot \|\underline{u}(j\omega)\|_2$$

$$\Rightarrow \boxed{\|\underline{y}(j\omega)\|_2 \leq \|G(j\omega)\|_\infty \cdot \|\underline{u}(j\omega)\|_2}$$

Similarly:

$$\|\underline{y}\|_s \leq \|\underline{\underline{G}}\|_\infty \cdot \|\underline{u}\|_s$$

$$\|\underline{y}\|_p \leq \|\underline{\underline{G}}\|_\infty \cdot \|\underline{u}\|_p$$

$$\|\underline{y}\|_p \leq \|\underline{\underline{G}}\|_2 \cdot \|\underline{u}\|_s$$

$$\|\underline{y}\|_\infty \leq \|\underline{\underline{G}}\|_2 \cdot \|\underline{u}\|_2$$

$$\|\underline{y}\|_\infty \leq \|\underline{\underline{G}}\|_1 \cdot \|\underline{u}\|_\infty$$

where:

$$\|\underline{\underline{G}}\|_1 \stackrel{!}{=} \max_i \underbrace{\|\underline{g}_i(t)\|_1}_{\text{signal-norm}}$$

(not currently implemented in Polpac).