This is the documentation for the program **DepthFirst.mws** written in Maple (version 13). The program is intended to justify claims made in an article submitted to LAA to establish a counterexample to a conjecture of Matsaev. This involves verifying that two matrices have ℓ^4 operator norms less than unity. The procedures in this program verify that this is the case. We go through contorsions to ensure that floating point arithmetic is nowhere used. All calculations involve rational numbers and are therefore free of any possible roundoff error.

The program should run on a modern PC in under 24 hours. The verification for the first matrix takes seconds only.

In principle, the program cound be used to generate the finite list of patches appearing in the hypotheses of Lemma 6 of the article.

```
> restart;
> with(linalg):
```

This procedure computes the fourth power of the ℓ^4 norm of a vector or list.

```
> L4NormToPower4:=proc(L)
> local sum,x,M;
> M:=convert(L,list);
> sum:=0;
> for x in M do sum:=sum+x^4;od;
> return(eval(sum));
> end;
```

$L4NormToPower4 := \mathbf{proc}(L)$
$\mathbf{local}\ sum,\ x,\ M;$
$M := \mathrm{convert}(L,\ list);$
$sum := 0;$
$\mathbf{for}\ x\ \mathbf{in}\ M\ \mathbf{do}\ sum := sum + x^4\ \mathbf{end\ do};$
$\mathbf{return}\ \mathrm{eval}(sum)$
$\mathbf{end\ proc}$

The procedure **verify_surface_subcube** takes a surface subcube as its argument specified by the centre, halfside and face in which it lies. The return value is

- 0 if the value of $\|T\rho(\xi)\|_4 > 1$ where ξ is the centre of the surface subcube.
- 2 if the surface subcube can be eliminated
- 1 otherwise. In this case, the surface subcube must be subdivided after the return.

```
> verify_surface_subcube:=proc(centre,halfside,face_index)
> global T;
> local s,n,y;
> n:=coldim(T);
```

We use the estimate for $\tilde{r}$ suggested in the article. This gives $1 - \frac{p-1}{2}\tilde{r}^2 = 1 - \frac{3}{2}(2n^{\frac{1}{4}}h)^2 = 1 - 6n^{\frac{1}{2}}h^2$ where h is the halfside. In order to avoid taking the fourth root, we use the inequality $1 - 24n^{\frac{1}{2}}h^2 \le (1 - 6n^{\frac{1}{2}}h^2)^4$ for elimination. If $1 - 24n^{\frac{1}{2}}h^2 \le 0$, then elimination will not be possible and we elect to subdivide.

```
> if (24^2*halfside^4*n >=1) then return(1);fi;
> y:=evalm(T &* centre);
> s:=L4NormToPower4(y)/L4NormToPower4(centre);
```

If $s > 1$, then $\|T\xi\|_4 > \|\xi\|_4$ where ξ is the centre of the surface subcube. We report failure.

```
> if (evalb(s > 1)) then return(0);fi;
```

We check the criterion for elimination so as to avoid computing $n^{\frac{1}{2}}$ explicitly.

```
> s:=evalb(24^2*n*halfside^4<=(1-s)^2);
```

```
> if s then return(2) else return(1);fi;
> end;
```

verify_surface_subcube := **proc**(*centre*, *halfside*, *face_index*)
local s, n, y;
global T;
n := *linalg* : −*coldim*(T);
if $1 \leq 576 * halfside^4 * n$ **then return** 1 **end if**;
y := evalm(‘& ∗ ‘(T, *centre*));
s := L4NormToPower4(y)/L4NormToPower4(*centre*);
if evalb($1 < s$) **then return** 0 **end if**;
s := evalb($576 * halfside^4 * n \leq (1 - s)^2$);
if s **then return** 2 **else return** 1 **end if**
end proc

The procedure **do_surface_subcube** takes a surface subcube as its argument specified by the centre, halfside and face in which it lies. The return value is either true or false. True is returned if the surface subcube can be subdivided into further surface subcubes all of which can be eliminated and False if at some point a surface subcube is encountered such that $\|T\xi\|_4 > \|\xi\|_4$ where ξ is its centre. It is possible for the program to hang in this procedure.

```
> do_surface_subcube:=proc(centre,halfside,face_index)
> global T;
> local n,h,k,kk,new_centre,i,j;
> k:=verify_surface_subcube(centre,halfside,face_index);
> if k=0 then return(false);fi;
> if k=2 then return(true);fi;
```

The following code subdivides the surface subcube dyadically and tests each of the offspring recursively. If any of the offspring report failure, then failure is reported. If all the offspring report success then success is reported.

```
> n:=coldim(T);
> h:=halfside/2;
> for k from 0 to 2^(n-1)-1 do
> kk:=k;
> new_centre:=[0$n];
> for j from 1 to n do
> if j=face_index then new_centre[j]:=centre[j];next;fi;
> new_centre[j]:=centre[j]+h*(2*irem(kk,2)-1);
> kk:=iquo(kk,2);
> od;
> if not do_surface_subcube(new_centre,h,face_index) then return(false);fi;
> od;
> return(true);
> end;
```

```
do_surface_subcube := proc(centre, halfside, face_index)
local n, h, k, kk, new_centre, i, j;
global T;
    k := verify_surface_subcube(centre, halfside, face_index);
    if k = 0 then return false end if;
    if k = 2 then return true end if;
    n := linalg : -coldim(T);
    h := 1/2 * halfside;
    for k from 0 to 2^(n-1) - 1 do
        kk := k;
        new_centre := [0 $ n];
        for j to n do
            if j = face_index then new_centre_j := centre_j; next end if;
            new_centre_j := centre_j + h * (2 * irem(kk, 2) - 1);
            kk := iquo(kk, 2)
        end do;
        if not do_surface_subcube(new_centre, h, face_index) then return false end if
    end do;
    return true
end proc
```

The procedure **verify_face** implements **do_surface_subcube** on the full face given by face_index

```
> verify_face:=proc(face_index)
> global T;
> local halfside,n,centre;
> n:=coldim(T);
> halfside:=1;
> centre:=[0$n];
> centre[face_index]:=1;
> return(do_surface_subcube(centre, halfside,face_index));
> end;
```

```
verify_face := proc(face_index)
local halfside, n, centre;
global T;
    n := linalg : -coldim(T);
    halfside := 1;
    centre := [0 $ n];
    centre_face_index := 1;
    return do_surface_subcube(centre, halfside, face_index)
end proc
```

The procedure **verify_all** performs **verify_face** on every face. As indicated in the article, it is only necessary to do this check on the faces that take the value 1 in their face_index coordinate. If any of the faces report failure, then failure is reported. If all the faces report success then success is reported.

```
> verify_all:=proc()
> global T;
> local n,j;
> n:=coldim(T);
```

```
> for j from 1 to n do if not verify_face(j) then return(false);fi;od;
> return(true);
> end;
```

```
verify_all := proc()
local n, j;
global T;
  n := linalg : -coldim(T);
  for j to n do if not verify_face(j) then return false end if end do;
  return true
end proc
```

In the remainder of the program we check the matrices outlined in the article.

```
> T:=matrix([[-292187/1000000,812218/1000000],[-958489/1000000,-113428/
1000000]]);
```

$$T := \begin{bmatrix} \frac{-292187}{1000000} & \frac{406109}{500000} \\ \frac{-958489}{1000000} & \frac{-28357}{250000} \end{bmatrix}$$

```
> verify_all();
```

$$true$$

```
> T:=array(1..7,1..5);
```

$$T := \mathrm{array}(1..7,\ 1..5,\ [])$$

```
> C:=652773/100000;
```

$$C := \frac{652773}{100000}$$

```
> c:=[1,2,-22/5];
```

$$c := [1,\ 2,\ \frac{-22}{5}]$$

```
> mu:=[263485/1000000,385467/1000000,360598/1000000,302274/1000000,1657
18/1000000];
```

$$\mu := [\frac{52697}{200000}, \frac{385467}{1000000}, \frac{180299}{500000}, \frac{151137}{500000}, \frac{82859}{500000}]$$

```
> nu:=[64388/1000000,146465/1000000,300904/1000000,364746/1000000,31852
5/1000000,247617/1000000,123827/1000000];
```

$$\nu := [\frac{16097}{250000}, \frac{29293}{200000}, \frac{37613}{125000}, \frac{182373}{500000}, \frac{12741}{40000}, \frac{247617}{1000000}, \frac{123827}{1000000}]$$

```
> for j from 1 to 7 do for k from 1 to 5 do if j-k>=0 and j-k<=2
then T[j,k]:=mu[k]*c[j-k+1]/nu[j]/C; else T[j,k]:=0;fi;od;od;
> print(T);
```

$$\begin{bmatrix} \frac{6587125000}{10507686981} & 0 & 0 & 0 & 0 \\ \frac{10539400000}{19121679489} & \frac{2569780000}{6373893163} & 0 & 0 & 0 \\ \frac{-1317425000}{2232068259} & \frac{3212225000}{8184250283} & \frac{4507475000}{24552750849} & 0 & 0 \\ 0 & \frac{-2569780000}{3607520313} & \frac{36059800000}{119048170329} & \frac{1679300000}{13227574481} & 0 \\ 0 & 0 & \frac{-576956800}{756089163} & \frac{268688000}{924108977} & \frac{662872000}{8316980793} \\ 0 & 0 & 0 & \frac{-1343440000}{1632703959} & \frac{33143600000}{161637691941} \\ 0 & 0 & 0 & 0 & \frac{-6628720000}{7348265661} \end{bmatrix}$$

```
> verify_all();
```

$$true$$