Analysis of Subdivision based on C^4 box spline

Denis Zorin, may 2000

Utilities

Matrix initialization and barycenter computations

Insert zero entries into the matrix: lazy subdivision, initialization for primal schemes.

```
> upsample := proc(M,i,j) if irem(i-1,2) = 0 and irem(j-1,2) = 0 then M[(i-1)/2+1,(j-1)/2+1]; else 0; fi; end;
```

upsample := **proc**(*M*, *i*, *j*) **if** irem(*i* − 1, 2) = 0 **and** irem(*j* − 1, 2) = 0 **then** *M*[1 / 2*i + 1 / 2, 1 / 2*j + 1 / 2] **else** 0 **fi end**

Primal to dual step: compute barycenters; shrinks the matrix size by 1 vertically. special treatment of top left entry, which depends on the center, which is provided as the second argument. Missing entries m[i,0] on the left replaced by omega*m[1,i-1]

```
> primal2dual := proc(M,g,i,j)
      if j <> 1
      then M[i,j-1]+M[i+1,j-1]+M[i,j]+M[i+1,j];
      elif i <> 1 then omega*(M[1,i-1]+M[1,i])+M[i,j]+M[i+1,j];
      else g+omega*M[1,1]+M[1,1]+M[2,1];
      fi;
  end;
```

primal2dual := **proc**(*M*, *g*, *i*, *j*)

if *j* ≠ 1 **then** *M*[*i*, *j* − 1] + *M*[*i* + 1, *j* − 1] + *M*[*i*, *j*] + *M*[*i* + 1, *j*] **elif** *i* ≠ 1 **then** ω*(*M*[1, *i* − 1] + *M*[1, *i*]) + *M*[*i*, *j*] + *M*[*i* + 1, *j*] **else** *g* + ω*(*M*[1, 1] + *M*[1, 1] + *M*[2, 1] **fi**

end

Dual to primal step: compute barycenters; decreases the matrix size by 1 horizontally. Missing entries m[0,j] on the top are replaced by m[j,1]

```
> dual2primal := proc(M,i,j)
    local m;
     if i <> 1  then
         M[i-1,j]+M[i-1,j+1]+M[i,j]+M[i,j+1];
     else
       (M[j,1]+M[j+1,1])*conjugate(omega) + M[i,j]+M[i,j+1];
     fi;
  end:
```

Two dualization steps taking a primal sector matrix to the next

```
> makeSectorMatrixPrimal := proc(M, g, steps) local m, size, Mdual, Mprimal;
    global dual2primal, primal2dual;
    Mprimal := M;
    size := rowdim(M);
    for m from 0 to steps-1 do
      Mdual := matrix(size-m-1,size-m-1,(i,j)->primal2dual(Mprimal,g,i,j));
      Mprimal := matrix(size-m-1,size-m-2, (i,j)->dual2primal(Mdual,i,j));
    od;
    evalm(Mprimal);
  end:
```

Subdivision matrix from sector matrix

This assumes N by N-1 matrix, which is numbered from the upper left corner 9see below for order illustration).

```
> renumberQuadPrimal := proc(i,j,size) local ni, nj, l, n;
     nj := j; ni := i-1;
     l := max(ni,nj);
     if( ni <= nj ) then n := i-1; else n := i+(i-j)-2; fi;
      l*(l-1) + n +1;
    end;
```

renumberQuadPrimal := **proc**(*i*, *j*, *size*) **local** *ni*, *nj*, *l*, *n*; *nj* := *j*; *ni* := *i* − 1; *l* := max(*ni*, *nj*); **if** *ni* ≤ *nj* **then** *n* := *i* − 1 **else** *n* := 2*i − *j* − 2 **fi**; *l*(*l* − 1) + *n* + 1 **end**

Illustration of reordaring:

```
> matrix( 7,6, (i,j)-> renumberQuadPrimal(i,j,5));
```

$$\begin{bmatrix} 1 & 3 & 7 & 13 & 21 & 31 \\ 2 & 4 & 8 & 14 & 22 & 32 \\ 6 & 5 & 9 & 15 & 23 & 33 \\ 12 & 11 & 10 & 16 & 24 & 34 \\ 20 & 19 & 18 & 17 & 25 & 35 \\ 30 & 29 & 28 & 27 & 26 & 36 \\ 42 & 41 & 40 & 39 & 38 & 37 \end{bmatrix}$$

Make DFT subdivision matrix out of a sector matrix, with order of entries given by renumber. This assumes that the initial sector matrix before subdivision had entries m[i,j]

```
> MakeSubdivMatrix := proc(M, renumber)
    local S, i,j,r,q;
    S := matrix(coldim(M)*rowdim(M), coldim(M)*rowdim(M));
    for i from  0 to rowdim(M)-1 do
      for j from  0 to coldim(M)-1 do
         for r from 0 to rowdim(M)-1 do
           for q from 0 to coldim(M)-1 do
              S[renumber(i+1, j+1,rowdim(M)),renumber(r+1,q+1,rowdim(M))] := coeff(M[i+1,j+1],m[r+1,q+1]);
           od;
         od;
      od;
    od;
   evalm(S);
  end:
```

Sligtly different version for omega=0

```
> MakeSubdivMatrixZero := proc(M, renumber)
```

```
    local S, i,j,r,q;
    S := matrix(coldim(M)*rowdim(M), coldim(M)*rowdim(M)+1);
    for i from  0 to rowdim(M)-1 do
      for j from  0 to coldim(M)-1 do
         S[renumber(i+1, j+1,rowdim(M)),1] := coeff(M[i+1,j+1],m[0,0]);
         for r from 0 to rowdim(M)-1 do
           for q from 0 to coldim(M)-1 do
              S[renumber(i+1, j+1,rowdim(M)),renumber(r+1,q+1,rowdim(M))+1] := coeff(M[i+1,j+1],m[r+1,q+1]);
           od;
         od;
      od;
    od;
   evalm(S);
  end:
```

\Subdivision matrix computation

Initial matrix size is N by N-1; after upsampling, it's size increases to 2*N-1 by 2*N-2 , after each pair of dualization steps, it decreases by 1 in each dimension.

```
N := 4;
```

$$N := 4$$

```
> M0 := matrix( N,N-1,(i,j) ->  m[i,j]  );
```

$$M0 := \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \\ m_{4,1} & m_{4,2} & m_{4,3} \end{bmatrix}$$

Upsample.

```
> M0 :=  matrix(2*N-1,2*N-2, (i,j)->upsample(M0,i,j-1));
```

$$M0 := \begin{bmatrix} 0 & m_{1,1} & 0 & m_{1,2} & 0 & m_{1,3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_{2,1} & 0 & m_{2,2} & 0 & m_{2,3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_{3,1} & 0 & m_{3,2} & 0 & m_{3,3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_{4,1} & 0 & m_{4,2} & 0 & m_{4,3} \end{bmatrix}$$

Two dualization steps result in midpoint subdivision. g is irrelevant in this case.

```
> SMmidpoint :=   map( simplify, subs(conjugate(omega) = 1/omega, makeSectorMatrixPrimal(M0, m[0,0],1 )));
```

$$SMmidpoint := \begin{bmatrix} \frac{m_{0,0} + 2\,\omega\, m_{1,1} + m_{0,0}\,\omega}{\omega} & 4\, m_{1,1} & 2\, m_{1,1} + 2\, m_{1,2} & 4\, m_{1,2} & 2\, m_{1,2} + 2\, m_{1,3} \\ m_{0,0} + m_{1,1} + \omega\, m_{1,1} + m_{2,1} & 2\, m_{1,1} + 2\, m_{2,1} & m_{1,1} + m_{1,2} + m_{2,1} + m_{2,2} & 2\, m_{1,2} + 2\, m_{2,2} & m_{1,2} + m_{1,3} + m_{2,2} + m_{2,3} \\ 2\,\omega\, m_{1,1} + 2\, m_{2,1} & 4\, m_{2,1} & 2\, m_{2,1} + 2\, m_{2,2} & 4\, m_{2,2} & 2\, m_{2,2} + 2\, m_{2,3} \\ \omega\, m_{1,1} + m_{2,1} + \omega\, m_{1,2} + m_{3,1} & 2\, m_{2,1} + 2\, m_{3,1} & m_{2,1} + m_{2,2} + m_{3,1} + m_{3,2} & 2\, m_{2,2} + 2\, m_{3,2} & m_{2,2} + m_{2,3} + m_{3,2} + m_{3,3} \\ 2\,\omega\, m_{1,2} + 2\, m_{3,1} & 4\, m_{3,1} & 2\, m_{3,1} + 2\, m_{3,2} & 4\, m_{3,2} & 2\, m_{3,2} + 2\, m_{3,3} \\ \omega\, m_{1,2} + m_{3,1} + \omega\, m_{1,3} + m_{4,1} & 2\, m_{3,1} + 2\, m_{4,1} & m_{3,1} + m_{3,2} + m_{4,1} + m_{4,2} & 2\, m_{3,2} + 2\, m_{4,2} & m_{3,2} + m_{3,3} + m_{4,2} + m_{4,3} \end{bmatrix}$$

Replace each point with the barycenter of edge neighbors; decreases the sector matrix by 1 in each dimension, assumes l by l-1 matrix; for entries on the top and on the left, missing 0 column and 0 row elements are replaced as
for primal2dual and dual2primal. Special treatment of m[1,1]: use the arg g as the center.

```
> baryDiag := proc(M,g,i,j)
    if i > 1 and j > 1
      then M[i-1,j]+M[i+1,j]+M[i,j+1]+M[i,j-1];
      elif i > 1 and j = 1 then M[i-1,j]+M[i+1,j]+M[i,j+1]+M[1,i-1]*omega;
      elif i = 1 and j > 1 then M[1,j-1]+M[1,j+1]+M[j+1,1]*conjugate(omega)+M[i+1,j];
      else g+M[1,j+1]+M[j+1,1]*conjugate(omega)+M[i+1,j];
      fi;
  end:
```

First filter step; the central vertex is unchanged by midoint subdivision; scale by 4 as we use unscaled coeffs for barycenters.

```
SMbox1 := matrix(rowdim(SMmidpoint)-1, coldim(SMmidpoint)-1, (i,j)->baryDiag(SMmidpoint,4*m[0,0],i,j)):
```

The effect of the previous step on the center is to replase it with the barycenter of edge neighbors, which is zero for omega <> 0, and (m[1,1]+m[0,0])/2 otherwise; again, scale by 4, but twice; d is 1 for zero freq. and 0 otherwise.

```
SMbox2 := matrix(rowdim(SMbox1)-1, coldim(SMbox1)-1, (i,j)->baryDiag(SMbox1,8*d*(m[1,1]+m[0,0]),i,j));
>
>
```

$$SMbox2 :=$$

$$\left[8\, d\, (m_{1,1} + m_{0,0}) + 2\, \frac{m_{0,0} + 2\,\omega\, m_{1,1} + m_{0,0}\,\omega}{\omega} + 6\, m_{1,1} + 2\, m_{1,2} + (2\,\omega\, m_{1,1} + 2\, m_{2,1})\, \overline{\omega} + 6\, m_{2,1} + \left(\frac{m_{0,0} + 2\,\omega\, m_{1,1} + m_{0,0}\,\omega}{\omega} + 4\,\omega\, m_{1,1} + 4\, m_{2,1} + 2\, m_{1,1} + m_{0,0} + m_{0,0}\,\omega \right) \overline{\omega} + 4\,\omega\, m_{1,1} \right.$$
$$\left. + m_{0,0} + m_{0,0}\,\omega \right.,$$
$$6\, m_{0,0} + 16\, m_{1,1} + (m_{0,0} + m_{1,1} + \omega\, m_{1,1} + m_{2,1})\, \overline{\omega} + 2\,\omega\, m_{1,1} + 8\, m_{2,1} + 6\, m_{1,2} + (\omega\, m_{1,1} + m_{2,1} + \omega\, m_{1,2} + m_{3,1})\, \overline{\omega} + 2\, m_{2,2} + (m_{0,0} + m_{1,1} + 6\,\omega\, m_{1,1} + 6\, m_{2,1} + \omega\, m_{1,2} + m_{3,1})\, \overline{\omega},$$
$$\left. \frac{m_{0,0} + 2\,\omega\, m_{1,1} + m_{0,0}\,\omega}{\omega} + 10\, m_{1,1} + 12\, m_{1,2} + (2\,\omega\, m_{1,1} + 2\, m_{2,1})\, \overline{\omega} + 6\, m_{2,1} + 2\, m_{1,3} + (2\,\omega\, m_{1,2} + 2\, m_{3,1})\, \overline{\omega} + 6\, m_{2,2} + (2\,\omega\, m_{1,1} + 4\, m_{2,1} + 2\,\omega\, m_{1,2} + 4\, m_{3,1} + \omega\, (2\, m_{1,1} + 2\, m_{1,2}))\, \overline{\omega} \right]$$

$$\left[7\, m_{0,0} + 12\, m_{1,1} + (m_{0,0} + m_{1,1} + \omega\, m_{1,1} + m_{2,1})\, \overline{\omega} + 8\,\omega\, m_{1,1} + 13\, m_{2,1} + \omega\, m_{1,2} + m_{3,1} + m_{1,2} + m_{2,2} + (5\, m_{0,0} + 5\, m_{1,1} + (m_{0,0} + m_{1,1} + \omega\, m_{1,1} + m_{2,1})\, \overline{\omega} + \omega\, m_{1,1} + m_{2,1})\, \omega \right.,$$
$$2\, \frac{m_{0,0} + 2\,\omega\, m_{1,1} + m_{0,0}\,\omega}{\omega} + 12\, m_{1,1} + 6\, m_{1,2} + (2\,\omega\, m_{1,1} + 2\, m_{2,1})\, \overline{\omega} + 18\, m_{2,1} + 2\, m_{3,1} + 6\, m_{2,2} + 6\,\omega\, m_{1,1} + m_{0,0} + m_{0,0}\,\omega \right.,$$
$$\left. 13\, m_{1,1} + 13\, m_{1,2} + (\omega\, m_{1,1} + m_{2,1} + \omega\, m_{1,2} + m_{3,1})\, \overline{\omega} + 14\, m_{2,1} + 14\, m_{2,2} + m_{3,1} + m_{3,2} + m_{1,3} + m_{2,3} + m_{0,0} + \omega\, m_{1,1} \right]$$

$$\left[\frac{m_{0,0}+2\,\omega\,m_{1,1}+m_{0,0}\,\omega}{\omega}+8\,\omega\,m_{1,1}+16\,m_{2,1}+4\,m_{1,1}+m_{0,0}+m_{0,0}\,\omega+2\,\omega\,m_{1,2}+6\,m_{3,1}+\omega\,(2\,m_{1,1}+2\,m_{1,2})+2\,m_{2,2}\right.$$
$$\left.+\left(\frac{m_{0,0}+2\,\omega\,m_{1,1}+m_{0,0}\,\omega}{\omega}+4\,m_{1,1}+2\,m_{1,2}+(2\,\omega\,m_{1,1}+2\,m_{2,1})\,\overline{\omega}+2\,m_{2,1}\right)\omega,\ 8\,m_{1,1}+24\,m_{2,1}+2\,m_{1,2}+8\,m_{2,2}+2\,m_{0,0}+8\,\omega\,m_{1,1}+8\,m_{3,1}+2\,m_{3,2}+2\,\omega\,m_{1,2},\right.$$
$$\left.6\,m_{1,1}+6\,m_{1,2}+18\,m_{2,1}+18\,m_{2,2}+6\,m_{3,1}+6\,m_{3,2}+2\,m_{2,3}+2\,\omega\,m_{1,1}\right]$$
$$[m_{0,0}+m_{1,1}+8\,\omega\,m_{1,1}+13\,m_{2,1}+8\,\omega\,m_{1,2}+13\,m_{3,1}+\omega\,m_{1,3}+m_{4,1}+m_{2,2}+m_{3,2}+(5\,m_{1,1}+5\,m_{1,2}+(\omega\,m_{1,1}+m_{2,1}+\omega\,m_{1,2}+m_{3,1})\,\overline{\omega}+m_{2,1}+m_{2,2})\,\omega,$$
$$2\,m_{1,1}+18\,m_{2,1}+18\,m_{3,1}+6\,m_{2,2}+4\,\omega\,m_{1,1}+2\,m_{4,1}+6\,m_{3,2}+4\,\omega\,m_{1,2}+\omega\,(2\,m_{1,1}+2\,m_{1,2}),$$
$$m_{1,1}+m_{1,2}+14\,m_{2,1}+14\,m_{2,2}+14\,m_{3,1}+14\,m_{3,2}+m_{4,1}+m_{4,2}+m_{2,3}+m_{3,3}+\omega\,m_{1,1}+\omega\,m_{1,2}]$$

```
> S := subs( d = 0, map( expand, map(simplify, subs( omega = cos(a) +I*sin(a),subs( conjugate(omega) = 1/omega,
  MakeSubdivMatrix(SMbox2, renumberQuadPrimal, false))))));
```

$$S := \begin{bmatrix}
16+8\cos(a), & -6\,I\sin(a)+6\cos(a)+6, & 2, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0\\
2\,I\cos(a)\sin(a)+2\cos(a)^2+13+13\,I\sin(a)+15\cos(a), & 14+2\cos(a), & 1+\cos(a)+I\sin(a), & 1, & 0, & 1, & 0, & 0, & 0, & 0, & 0, & 0\\
24+4\cos(a), & -8\,I\sin(a)+8\cos(a)+8, & 8, & 2, & 0, & -2\,I\sin(a)+2\cos(a), & 0, & 0, & 0, & 0, & 0, & 0\\
18+6\cos(a)+6\,I\sin(a), & -2\,I\sin(a)+2\cos(a)+18, & 6, & 6, & 0, & 2, & 0, & 0, & 0, & 0, & 0, & 0\\
8+8\cos(a)+8\,I\sin(a), & 24, & 2+2\cos(a)+2\,I\sin(a), & 8, & 2, & 8, & 0, & 0, & 0, & 0, & 0, & 0\\
6+18\cos(a)+18\,I\sin(a), & 2\,I\sin(a)+18+2\cos(a), & 6\cos(a)+6\,I\sin(a), & 2, & 0, & 6, & 0, & 0, & 0, & 0, & 0, & 0\\
18, & -6\,I\sin(a)+6\cos(a)+6, & 18, & 6, & 0, & -6\,I\sin(a)+6\cos(a), & 2, & 0, & 0, & 0, & 0, & 0\\
14+\cos(a)+I\sin(a), & -I\sin(a)+\cos(a)+14, & 14, & 14, & 1, & -I\sin(a)+\cos(a)+1, & 1, & 1, & 0, & 0, & 0, & 0\\
6+2\cos(a)+2\,I\sin(a), & 18, & 6, & 18, & 6, & 6, & 0, & 2, & 0, & 0, & 0, & 0\\
1+\cos(a)+I\sin(a), & 14, & 1+\cos(a)+I\sin(a), & 14, & 14, & 14, & 0, & 1, & 1, & 0, & 1, & 1\\
2+6\cos(a)+6\,I\sin(a), & 18, & 6\cos(a)+6\,I\sin(a), & 6, & 6, & 18, & 0, & 0, & 0, & 0, & 0, & 2\\
1+14\cos(a)+14\,I\sin(a), & 14+\cos(a)+I\sin(a), & 14\cos(a)+14\,I\sin(a), & 1+\cos(a)+I\sin(a), & 1, & 14, & \cos(a)+I\sin(a), & 0, & 0, & 0, & 0, & 1
\end{bmatrix}$$

```
> subs( omega^2*conjugate(omega) = omega, subs( omega*conjugate(omega) = 1, map( expand, MakeSubdivMatrix(SMbox2, renumberQuadPrimal,
  false))));
```

$$\begin{bmatrix}
8\,d+16+4\,\overline{\omega}+4\,\omega & 6\,\overline{\omega}+6 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
14+\overline{\omega}+14\,\omega+\omega^2 & \overline{\omega}+14+\omega & 1+\omega & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
24+2\,\overline{\omega}+2\,\omega & 8\,\overline{\omega}+8 & 8 & 2 & 0 & 2\,\overline{\omega} & 0 & 0 & 0 & 0 & 0 & 0\\
18+6\,\omega & 2\,\overline{\omega}+18 & 6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0\\
8+8\,\omega & 24 & 2+2\,\omega & 8 & 2 & 8 & 0 & 0 & 0 & 0 & 0 & 0\\
6+18\,\omega & 18+2\,\omega & 6\,\omega & 2 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0\\
18 & 6\,\overline{\omega}+6 & 18 & 6 & 0 & 6\,\overline{\omega} & 2 & 0 & 0 & 0 & 0 & 0\\
14+\omega & \overline{\omega}+14 & 14 & 14 & 1 & \overline{\omega}+1 & 1 & 1 & 0 & 0 & 0 & 0\\
6+2\,\omega & 18 & 6 & 18 & 6 & 6 & 0 & 2 & 0 & 0 & 0 & 0\\
1+\omega & 14 & 1+\omega & 14 & 14 & 14 & 0 & 1 & 1 & 0 & 1 & 1\\
2+6\,\omega & 18 & 6\,\omega & 6 & 6 & 18 & 0 & 0 & 0 & 0 & 0 & 2\\
1+14\,\omega & 14+\omega & 14\,\omega & 1+\omega & 1 & 14 & \omega & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

```
> SZero := map( expand, map(simplify, subs(  {d = 1, omega = 1},subs( conjugate(omega) = 1/omega,  MakeSubdivMatrixZero(SMbox2,
  renumberQuadPrimal, true))));
```

$$SZero := \begin{bmatrix}
18 & 32 & 12 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
14 & 30 & 16 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
8 & 28 & 16 & 8 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0\\
6 & 24 & 20 & 6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 & 0\\
2 & 16 & 24 & 4 & 8 & 2 & 8 & 0 & 0 & 0 & 0 & 0\\
6 & 24 & 20 & 6 & 2 & 0 & 6 & 0 & 0 & 0 & 0 & 0\\
2 & 18 & 12 & 18 & 6 & 0 & 6 & 2 & 0 & 0 & 0 & 0\\
1 & 15 & 15 & 14 & 14 & 1 & 2 & 1 & 1 & 0 & 0 & 0\\
0 & 8 & 18 & 6 & 18 & 6 & 6 & 0 & 2 & 0 & 0 & 0\\
0 & 2 & 14 & 2 & 14 & 14 & 14 & 0 & 1 & 1 & 0 & 1\\
0 & 8 & 18 & 6 & 6 & 6 & 18 & 0 & 0 & 0 & 0 & 2\\
1 & 15 & 15 & 14 & 2 & 1 & 14 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}$$

```
> SZero := stackmatrix( matrix( [[24,32,8,0,0,0,0,0,0,0,0,0]]),SZero);
```

$$SZero := \begin{bmatrix}
24 & 32 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
18 & 32 & 12 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
14 & 30 & 16 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
8 & 28 & 16 & 8 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0\\
6 & 24 & 20 & 6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 & 0\\
2 & 16 & 24 & 4 & 8 & 2 & 8 & 0 & 0 & 0 & 0 & 0\\
6 & 24 & 20 & 6 & 2 & 0 & 6 & 0 & 0 & 0 & 0 & 0\\
2 & 18 & 12 & 18 & 6 & 0 & 6 & 2 & 0 & 0 & 0 & 0\\
1 & 15 & 15 & 14 & 14 & 1 & 2 & 1 & 1 & 0 & 0 & 0\\
0 & 8 & 18 & 6 & 18 & 6 & 6 & 0 & 2 & 0 & 0 & 0\\
0 & 2 & 14 & 2 & 14 & 14 & 14 & 0 & 1 & 1 & 0 & 1\\
0 & 8 & 18 & 6 & 6 & 6 & 18 & 0 & 0 & 0 & 0 & 2\\
1 & 15 & 15 & 14 & 2 & 1 & 14 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}$$

```
> map( x -> x/64, [eigenvals(SZero)]);
```

$$\left[\frac{1}{4}, 1, \frac{1}{64}, \frac{1}{32}, \frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, 0, 0, 0, 0\right]$$

```
> unassign('S');
> lEV := subs( { cos(a) = c,  sin(a) = s, _t[1]= 1},simplify( linsolve( submatrix( evalm( (1/64)*transpose(S) - lambda *
  diag(seq(1,i=1..12))), 2..12,1..12), [seq(0,i=1..11)])));
```

$$lEV := \left[\frac{1}{8}\,\frac{16-1120\,\lambda+9\,c+21248\,\lambda^2+c^2-139264\,\lambda^3+262144\,\lambda^4-8192\,\lambda^3\,c-192\,\lambda\,c+1280\,\lambda^2\,c}{-1+32\,\lambda-320\,\lambda^2-4\,\lambda\,c+64\,\lambda^2\,c+1024\,\lambda^3},\right.$$
$$\left.-\frac{1}{2}\,\frac{-1664\,I\,\lambda^2\,s-6144\,\lambda^3\,c-2\,I\,s+1664\,\lambda^2\,c-128\,\lambda\,c+3\,c+c^2-6144\,\lambda^3+1664\,\lambda^2-128\,\lambda+6144\,I\,\lambda^3\,s-I\,c\,s+128\,I\,\lambda\,s+2}{-1+32\,\lambda-320\,\lambda^2-4\,\lambda\,c+64\,\lambda^2\,c+1024\,\lambda^3}, 1,\right.$$

$$\frac{1}{16}\frac{-224\,\lambda+1280\,\lambda^2+96\,I\,\lambda\,s-64\,\lambda\,c+c+8+c^2-I\,c\,s+768\,\lambda^2\,c-768\,I\,\lambda^2\,s}{-1+32\,\lambda-320\,\lambda^2-4\,\lambda\,c+64\,\lambda^2\,c+1024\,\lambda^3},\ 0,$$

$$-\frac{1}{16}\frac{-9\,c+8\,I\,s+224\,\lambda\,c-1280\,\lambda^2\,c-32\,\lambda\,c^2+32\,I\,\lambda\,s\,c-c^2+96\,\lambda-768\,\lambda^2+1280\,I\,\lambda^2\,s+I\,c\,s-224\,I\,\lambda\,s}{-1+32\,\lambda-320\,\lambda^2-4\,\lambda\,c+64\,\lambda^2\,c+1024\,\lambda^3},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\Big]$$

Eigenvalues and eigenvectors

```
> CharPoly := collect( simplify( charpoly(S, lambda)), lambda);
```

$$CharPoly := \lambda^{12}+(-10\cos(a)-56)\,\lambda^{11}+(985-8\cos(a)^2+196\cos(a))\,\lambda^{10}+(272\cos(a)^2-7894-1362\cos(a))\,\lambda^9+(-1960\cos(a)^2+3872\cos(a)+32636)\,\lambda^8$$
$$+(-1960\cos(a)+8640\cos(a)^2+512\cos(a)^3-74440)\,\lambda^7+(-24736\cos(a)^2-3072\cos(a)^3+94336-13408\cos(a))\,\lambda^6+(31104\cos(a)-61952+6656\cos(a)^3+40320\cos(a)^2)\,\lambda^5$$
$$+(16384-6144\cos(a)^3-26624\cos(a)-32768\cos(a)^2)\,\lambda^4+(2048\cos(a)^3+8192\cos(a)+10240\cos(a)^2)\,\lambda^3$$

```
>
> CharPoly := factor( subs( cos(a) = c, CharPoly));
```

$$CharPoly := (\lambda-2)^2\,(\lambda-1)^2\,\lambda^3\,(\lambda^5-10\,\lambda^4\,c-50\,\lambda^4+672\,\lambda^3-8\,\lambda^3\,c^2+136\,\lambda^3\,c+224\,\lambda^2\,c^2-3200\,\lambda^2-416\,\lambda^2\,c-512\,\lambda\,c^2-512\,\lambda\,c+4096\,\lambda+2048\,c+2560\,c^2+512\,c^3)$$

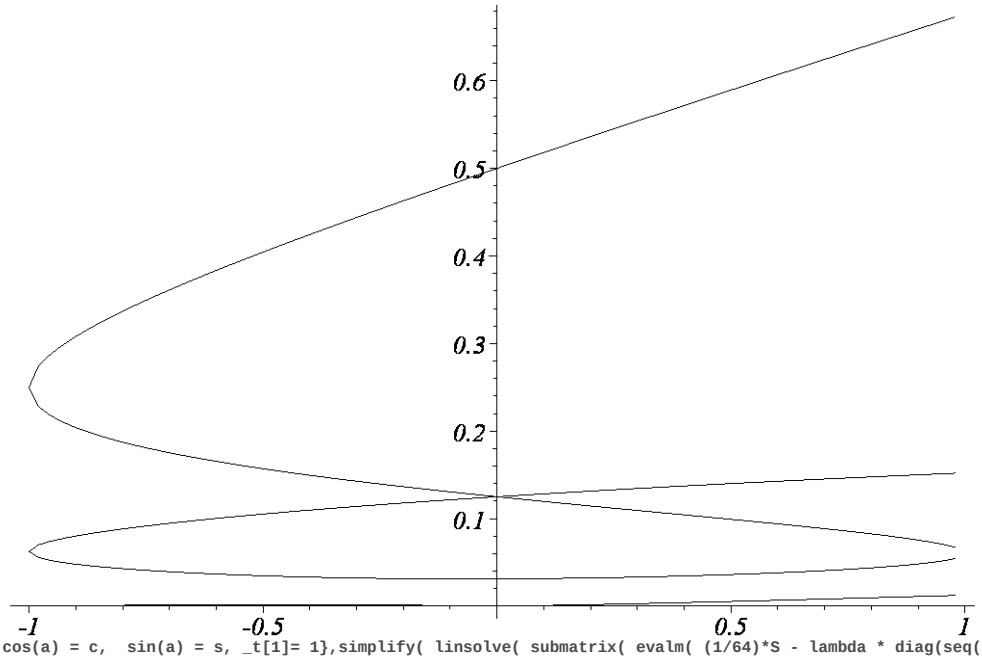
```
> CharPolyMainFactor := collect( subs( lambda = 64*lambda,
  lambda^5-50*lambda^4-10*lambda^4*c+672*lambda^3+136*lambda^3*c-8*lambda^3*c^2+224*lambda^2*c^2-3200*lambda^2-416*lambda^2*c+4096*lam
  bda-512*lambda*c^2-512*lambda*c+512*c^3+2048*c+2560*c^2), lambda);
>
```

$$CharPolyMainFactor := 1073741824\,\lambda^5+(-838860800-167772160\,c)\,\lambda^4+(35651584\,c+176160768-2097152\,c^2)\,\lambda^3+(917504\,c^2-13107200-1703936\,c)\,\lambda^2$$
$$+(-32768\,c+262144-32768\,c^2)\,\lambda+2048\,c+2560\,c^2+512\,c^3$$

```
> CharPolyMainFactor := collect( CharPolyMainFactor/64^5, lambda);
```

$$CharPolyMainFactor := \lambda^5+\left(-\frac{25}{32}-\frac{5}{32}c\right)\lambda^4+\left(\frac{17}{512}c+\frac{21}{128}-\frac{1}{512}c^2\right)\lambda^3+\left(\frac{7}{8192}c^2-\frac{25}{2048}-\frac{13}{8192}c\right)\lambda^2+\left(-\frac{1}{32768}c+\frac{1}{4096}-\frac{1}{32768}c^2\right)\lambda+\frac{1}{524288}c+\frac{5}{2097152}c^2+\frac{1}{2097152}c^3$$

```
> latex(CharPolyMainFactor);
{\lambda}^{5}+\left (-{\frac {25}{32}}-{\frac {5}{32}}\,c\right ){
\lambda}^{4}+\left ({\frac {17}{512}}\,c+{\frac {21}{128}}-{\frac {1}{
512}}\,{c}^{2}\right ){\lambda}^{3}+\left ({\frac {7}{8192}}\,{c}^{2}-
{\frac {25}{2048}}-{\frac {13}{8192}}\,c\right ){\lambda}^{2}+\left (-
{\frac {1}{32768}}\,c+{\frac {1}{4096}}-{\frac {1}{32768}}\,{c}^{2}
\right )\lambda+{\frac {1}{524288}}\,c+{\frac {5}{2097152}}\,{c}^{2}+{
\frac {1}{2097152}}\,{c}^{3}
> NumPts := 100: EigenvalsList := seq( sort(map( abs, [solve( subs( c = evalf( 2*(n+1e-10)/NumPts),CharPolyMainFactor))])), n =
  -NumPts/2..NumPts/2):
> EigenvalsPlotLists := seq( [ seq( [-1 + 2*(i-1)/NumPts, abs(op(j, op(i,[EigenvalsList])))], i = 1..NumPts)] ,j=1..5):
> display(seq(plot(op(i,[EigenvalsPlotLists]),color=black), i = 1..5),color=black, axesfont=[TIMES,ITALIC,10], labels=[‘‘,‘‘]);
```




```
> EV := subs( { cos(a) = c,  sin(a) = s, _t[1]= 1},simplify( linsolve( submatrix( evalm( (1/64)*S - lambda * diag(seq(1,i=1..12))),
  2..12,1..12), [seq(0,i=1..11)])));
> EVRe := map (simplify, map (evalc, map (Re, EV ))):
> EVIm := map (simplify, map( x -> x/s, map (evalc, map (Im, EV )))):
```

Eigenvalue analysis and interval estimation

Eigenvalues cannot be found explicitly (degree 5 equation). Here obtain information about the eigenvalues to verify C1-continuity. We prove that for any m, k, $m = 1 .. k-1$ the largest eigenvalue is real and unique, and that for $m \neq k-1$, 1 the largest eigenvalue is less than the largest eigenvalue of blocks 1 and $k-1$. We also show that the unique largest eigenvalue is a single eigenvalue in the interval [0.47+0.2c, 0.52+0.2c], for $k > 3$, where $c = \cos\left(\frac{2\pi}{k}\right)$

For $k = 3$, eigenvalues are examined separately. The proof is performed in several steps:

(1). We show that for c < 0, all roots of the characteristic polynomial $P(c, \lambda)$ are less than 0.51 (actually, they are less than 0.5, but due to numerical nature of our calculations, we have to relax the upper boundary).

(2). We show that for any $c = 0 .. 1$, there is a unique real root μ in the interval [0.47+0.2c, 0.5+0.2c], and the function $\mu(c)$ is C1-continuous and increases.
(3). We "deflate" the characteristic polynomial (that is, divide by the monomial $(\lambda - \mu)$ in symbolic form, with μ and c as indeterminates. Next, we verify that for all $c = 0 .. 1$, and $\mu = .47 + .2\,c .. .52 + .2\,c$, all roots of the deflated polynomial are inside the circle of radius 0.5 centered at 0 in the complex plane, that is, have magnitudes less than $\mu(c)$ for any $0 < c$.
As for $k > 4$, $.51 < \cos\left(\frac{2\pi}{k}\right)$ the largest eigenvalue cannot possibly correspond to a block m, for which $\cos\left(\frac{2m\pi}{k}\right) \le 0$. From (3), it follows that the largest root has to be the real root $\mu(c)$ for some c.
As for any $1 < m$, $m < k - 1$, $\cos\left(\frac{2m\pi}{k}\right) < \cos\left(\frac{2\pi}{k}\right)$, and we have shown (1) that $\mu(c)$, increases, and for any c $\mu(c)$ is the largest root, we conclude that the largest eigenvalue always corresponds to m = 1, is real, and is the unique eigenvalue in the range [0.47+0.2c, 0.52+0.2c].

On steps 1 and 3 we have to show that roots of a polynomial are inside a circle of radius r in the complex plane. This task is similar to the task of establishing stability of a filter with the transfer function $\frac{1}{a(z)}$, where $a(z)$ is a polynomial. Such filter is stable, if all roots of the polynomial are inside the unit circle.
A variety of tests exist for this condition; for our purposes, the algebraic Marden-Jury test is convenient. With aproporiate rescaling of the variable it can be used to prove that all roots of a polynomial are inside the circle of any given radius r. As the test requires only a simple algebraic calculaion on the coefficients of the polynomial, it can be easily performed for symbolic and interval coefficients.
Finally, we compute the largest root of the characteristic polynomial numerically for all valences up to some maximum. For each computed root, we verify that that the precision is at least $\varepsilon = .1\ 10^{-10}$: we use interval arithmetics to evaluate the polynomial at $\lambda_0 - \varepsilon$ and $\lambda_0 + \varepsilon$ and assert that the sign is guaranteed to change. There may be more than one root: we still have to prove that there is only a single root in the computed interval and that the rest of the roots are smaller. The maximal valence N is chosen in such a way that for N $\cos\left(\frac{2\pi}{N}\right)$ is "sufficiently close" to 1. This means that for all K > N corresponding eigenvalue differs from the limit value λ_∞ by no more than ε, where ε is small enough for us to establish, using interval arithmetics, that the Jacobian of the characteristic map is positive for eigenvectors computed using formulas derived below for all λ in the interval $[\lambda_\infty - \varepsilon, \lambda_\infty]$. The actual computation of the Jacobian and evaluation of the necessary contraction functions is performed in the C part of the code (This is not done yet; the value of ε can be detrmined only after the C code is made to to work with dual schemes).

Marden-Jury test

`MardenJury(a,var,rootrad)` Compute Marden-Jury table for a polynomial **`p`** in variable **`var`**, with variable rescaled by rootrad.
Used to verify that all roots of a polynomial are inside the circle of radius r.

```
> MardenJury := proc(a::polynom, var::name, rootrad)
  local i, k, M, acol, tbl, restable;
      M := degree(a, var);
      for i from 0 to M do tbl[0, i] := coeff(a, var, i)*rootrad^(i - M) od;
      for i to M do for k from 0 to M - i do tbl[i, k] := tbl[i - 1, 0]*tbl[i - 1, k] - tbl[i - 1, M - k - i + 1]*tbl[i - 1, M - i + 1] od od;
      for i to M do restable[i] := tbl[i, 0] od;
      eval(restable)
  end
```

Interval version of Marden-Jury test

```
> IntervMardenJury := proc(a::polynom(interval), var::name, rootrad::numeric)
  local i, k, M, acol, tbl, restable;
      M := degree(a, var);
      for i from 0 to M do tbl[0, i] := Interval_times(coeff(a, var, i), rootrad^(i - M)) od;
      for i to M do
          for k from 0 to M - i do tbl[i, k] := Interval_add(Interval_times(tbl[i - 1, 0], tbl[i - 1, k]), Interval_times(-1, Interval_times(tbl[i - 1, M - k - i + 1], tbl[i - 1, M - i + 1]))) od
      od;
      for i to M do restable[i] := tbl[i, 0] od;
      eval(restable)
  end
>
```

Deflation

`deflate(p,var,rootval)` compute the coefficients of the polynomial $\frac{p(z)}{z - z_0}$, it is assumed that p is divisible by $z - z_0$. **`var`** is the name of the variable, **`rootval`** is the root.

```
> deflate :=
  proc(p::polynom, var::name, rootval) local i, dp, r; dp := 0; r := lcoeff(p, var); for i from degree(p, var) - 1 by -1 to 0 do dp := dp + r*var^i; r := coeff(p, var, i) + rootval*r od; dp end
```

Analysis of the eigenvalues

Now we perform steps 1-3 described above.

(1). We show that for c < 0, all roots of the characteristic polynomial $P(c, \lambda)$ are less than 0.51 (actually, they are less than 0.5, but due to numerical nature of our calculations, we have to relax the upper boundary).

```
> 
> MJtab := MardenJury(CharPolyMainFactor, lambda, 51/100):
> MJtabInterv := map( unapply( 'inapply'( dummy, c), dummy), MJtab ):
> TestNegativeC := proc(cstart::numeric, cend::numeric, cstep::numeric)
  local MJ, cx;
  global MJtabInterv;
      for cx from cstart by cstep to cend do
          MJ := map(unapply(dummy([cx, cx + cstep]), dummy), MJtabInterv);
          if 0 < op(2, MJ[1]) or op(1, MJ[2]) < 0 or op(1, MJ[3]) < 0 or op(1, MJ[4]) < 0 then ERROR('test failed for interval = ', [cx, cx + cstep]) fi
      od;
      print('All tests passed')
  end
>
```

Do tests, adjusting the step in c. This is not really necessary -- we can simply take the smallest step, but to save time we use larger steps first.

```
> TestNegativeC(-1.0,-0.65,0.05);
```

All tests passed

```
> TestNegativeC(-0.6,-0.275,0.025);
```

All tests passed

```
> TestNegativeC(-0.25,-0.06,0.01);
```

All tests passed

```
> TestNegativeC(-0.05,0.,0.005);
```

All tests passed

(2). We show that for any $c = 0 .. 1$, there is a unique real root μ in the interval [0.47+0.2*c,0.52+0.2*c], and the function $\mu(c)$ increases.

Derivative $\frac{\partial}{\partial c}\lambda$ can be computed as $-\frac{\frac{\partial}{\partial c}F}{\frac{\partial}{\partial \lambda}F}$ if F(λ, c) is the char. polynomial. We use the derivative only to show that λ(c) increases, thus, we need only the sign. We will see that in the domain of interest the numerator is always negative, thus we need to look only on the sign of $\frac{\partial}{\partial \lambda}F$

```
> Flambda := diff( CharPolyMainFactor, lambda );
```

$$Flambda := 5\lambda^4 + 4\left(-\frac{25}{32} - \frac{5}{32}c\right)\lambda^3 + 3\left(\frac{17}{512}c + \frac{21}{128} - \frac{1}{512}c^2\right)\lambda^2 + 2\left(\frac{7}{8192}c^2 - \frac{25}{2048} - \frac{13}{8192}c\right)\lambda - \frac{1}{32768}c + \frac{1}{4096} - \frac{1}{32768}c^2$$

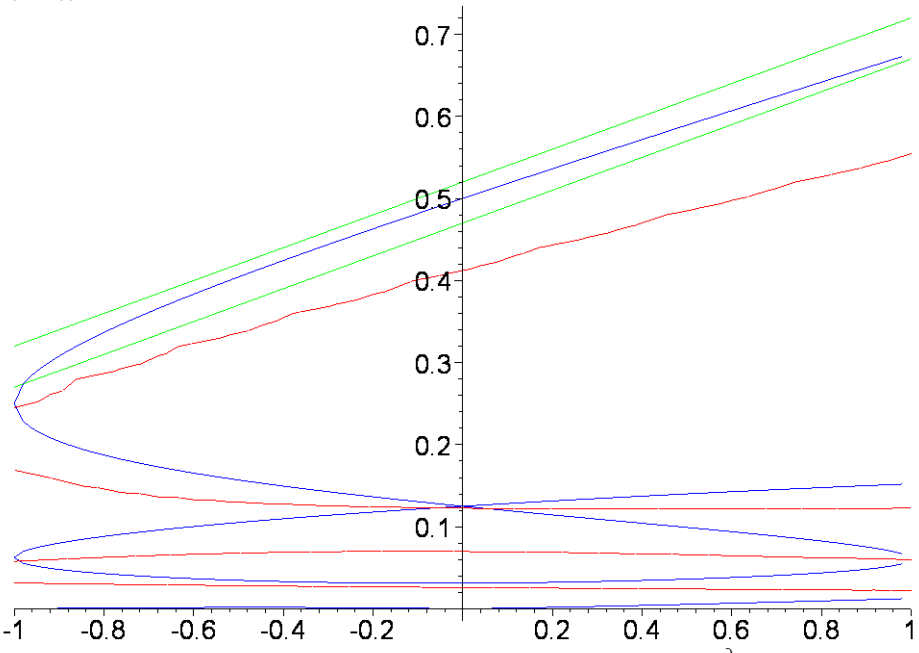
```
> Fc := diff( CharPolyMainFactor, c);
```

$$Fc := -\frac{5}{32}\lambda^4 + \left(\frac{17}{512} - \frac{1}{256}c\right)\lambda^3 + \left(\frac{7}{4096}c - \frac{13}{8192}\right)\lambda^2 + \left(-\frac{1}{32768} - \frac{1}{16384}c\right)\lambda + \frac{1}{524288} + \frac{5}{1048576}c + \frac{3}{2097152}c^2$$

```
>
```

This plot illustrates the idea: for c > 0, the largest root can be bounded from above and below by linear functions of c (green lines). The area between the lines is above the red curve indicating the zero set of $\frac{\partial}{\partial \lambda}F$, where derivative is positive.

```
> display( contourplot( unapply( Flambda,[c, lambda]), -1..1, 0..1, contours=[0], color=red),
  seq(plot(op(i,[EigenvalsPlotLists]),color=blue), i = 1..5), plot( 0.47+0.2*c, c = -1..1, color=green), plot( 0.52+0.2*c, c =
  -1..1, color = green));
```



To prove rigorously that there is a single root between the green lines, and that it increases as a function of c, we compute the derivative $\frac{\partial}{\partial \lambda}F$ in an area covering the area between the lines, verify that it is positive, and verify that the characteristic polynomial has different signs on the two lines. All calculations are done in interval arithmetic.

```
> VerifyMaxEvBounds := proc( charpoly, lowOffset::numeric, highOffset::numeric, slope::numeric,
                             csteps::integer, lambdasteps::integer, crange)
    local i,j, deltac, deltalambda, charpolyLowI, charpolyHighI,
         deriv, derivI, curDeriv, curCharHigh, curCharLow, d, failflag, curCrange, curLambdaRange;
    deltac := (crange[2]-crange[1])/csteps;
    deltalambda := (highOffset- lowOffset)/lambdasteps;
    charpolyLowI := subs( lambda = lowOffset + slope*(c-crange[1]), charpoly);
    charpolyLowI := inapply(charpolyLowI, c );
    charpolyHighI := subs( lambda = highOffset + slope*(c-crange[1]), charpoly);
    charpolyHighI := inapply(charpolyHighI, c );
    deriv := diff( charpoly, lambda );
    derivI := subs( lambda = d + slope*(c-crange[1]), deriv);
    derivI := inapply(derivI, [c,d] );
    failflag := false;

    for i from 0 to csteps-1 do
      curCrange := map(evalf, [crange[1]+ deltac*i, crange[1]+ deltac*(i+1)]);
      curCharHigh := charpolyHighI( curCrange);
      curCharLow  := charpolyLowI(  curCrange);
      if curCharHigh[1]*curCharHigh[2] <= 0 or
          curCharLow[1]*curCharLow[2] <= 0 or
          curCharLow[2]*curCharHigh[1] >= 0 then
        failflag := true;
        print('Sign change test failed for interval', curCrange,curCharLow,curCharHigh);
      fi;
      for j from 0 to lambdasteps-1 do
        curLambdaRange := map(evalf, [lowOffset+j*deltalambda+(curCrange[1]-crange[1])*slope,
                                      lowOffset+(j+1)*deltalambda+(curCrange[1]-crange[1])*slope]);
        curDeriv := derivI( curCrange, curLambdaRange);
        if curDeriv[1]*curDeriv[2] <= 0 or curDeriv[1] < 0 then
          print('Derivative may be not positive for c,lambdain the ranges', curCrange, curLambdaRange);
          failflag := true;
        fi;
```

```
      od;
      if i mod 10 = 0 then print(i);fi;
    od;
    if not failflag then print( 'All tests succeeded'); fi;
  end:
>
> VerifyMaxEvBounds( CharPolyMainFactor, 47/100, 52/100, 20/100,
                              160, 10, [0,85/100]);
```

0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150

Sign change test failed for interval, [.8181250000, .8234375000], [-.002295345401, .1084500001 10^{-5}], [.004081707299, .007037496101]
Sign change test failed for interval, [.8234375000, .8287500000], [-.002296706101, .00001363680001], [.004132990999, .007105592301]
Sign change test failed for interval, [.8287500000, .8340625000], [-.002297910201, .00002640860001], [.004184755599, .007174240701]
Sign change test failed for interval, [.8340625000, .8393750000], [-.002298956101, .00003940220001], [.004237003599, .007243444501]
Sign change test failed for interval, [.8393750000, .8446875000], [-.002299841601, .00005261960001], [.004289738799, .007313207401]
Sign change test failed for interval, [.8446875000, .8500000000], [-.002300564901, .00006606280001], [.004342964199, .007383532701]

```
> VerifyMaxEvBounds( CharPolyMainFactor, 47/100+17/100, 52/100+17/100, 20/100,
                              50, 10, [85/100,1]);
>
```

0
10
20
30
40

All tests succeeded

```
>
```

(3). We "deflate" the characteristic polynomial (that is, divide by the monomial $\lambda - \mu$) in the symbolic form, with μ and c as the indeterminates. Next, we verify that for all $c = 0 .. 1$, and for all $\lambda = .5 .. .613$, all roots of the deflated polynomial are inside the circle of radius 0.5 centered at 0 in the complex plane, that is, have magnitudes less than $\mu(c)$ for any $0 < c$.

Symbolic deflation; if we substitute a pair c, $\mu(c)$ we get the deflated polynomial for a specific value of c. Normalize deflated polynomial.

```
> deflatedCharPoly := collect( expand( deflate(CharPolyMainFactor, lambda, mu)), lambda)/64^5;
```

$$deflatedCharPoly := \frac{1}{1073741824}\lambda^4 + \frac{1}{1073741824}\left(-\frac{25}{32} - \frac{5}{32}c + \mu\right)\lambda^3 + \frac{1}{1073741824}\left(\frac{17}{512}c + \frac{21}{128} - \frac{1}{512}c^2 - \frac{25}{32}\mu - \frac{5}{32}\mu\, c + \mu^2\right)\lambda^2$$
$$+ \frac{1}{1073741824}\left(\frac{7}{8192}c^2 - \frac{25}{2048} - \frac{13}{8192}c + \frac{17}{512}\mu\, c + \frac{21}{128}\mu - \frac{1}{512}\mu\, c^2 - \frac{25}{32}\mu^2 - \frac{5}{32}\mu^2\, c + \mu^3\right)\lambda - \frac{1}{35184372088832}c - \frac{1}{549755813888}\mu^2\, c^2 + \frac{1}{4398046511104} - \frac{25}{2199023255552}\mu$$
$$+ \frac{1}{1073741824}\mu^4 + \frac{7}{8796093022208}\mu\, c^2 - \frac{5}{34359738368}\mu^3\, c - \frac{1}{35184372088832}c^2 - \frac{13}{8796093022208}\mu\, c + \frac{17}{549755813888}\mu^2\, c + \frac{21}{137438953472}\mu^2 - \frac{25}{34359738368}\mu^3$$

Verify that for all c in 0..1 and μ in [0.47+0.2*c, 0.52+0.2*c] the deflated polynomial has roots of magnitude < 0.5

```
> TestDeflated :=  proc(csteps::integer, lambdasteps::integer, highOffset, lowOffset, slope, crange )
    local cf,i,j,m, MJ, cinterv, deflatedinterv, lambdainterv, deltac, deltalambda;
    deltac := (crange[2]-crange[1])/csteps;
    deltalambda := (highOffset- lowOffset)/lambdasteps;
    for i from 0 to 3 do
      cf[i] := inapply(  coeff(deflatedCharPoly,lambda,i), c, mu);
    od;
    for i from 0 to csteps-1 do
      cinterv := [ crange[1]+i*deltac, crange[1]+(i+1)*deltac];
      for j from 0 to lambdasteps-1 do
        lambdainterv := [ lowOffset + (cinterv[1]-crange[1])*slope + j*deltalambda,
                          lowOffset + (cinterv[2]-crange[1])*slope + (j+1)*deltalambda];
        deflatedinterv := 0;
        for m from 0 to 2 do
          deflatedinterv := deflatedinterv + eval(cf[m](cinterv,lambdainterv))*lambda^m;
        od;
        # fix for cf[3]: stupid inapply does not always return an interval
        deflatedinterv := deflatedinterv + [1.0,1.0]*lambda^3;
        MJ := IntervMardenJury( deflatedinterv, lambda, 0.5);
        if op(2, MJ[1]) > 0 or op(1, MJ[2]) < 0 or op(1,MJ[3]) < 0
         then ERROR('test failed for interval ', cinterv);
        fi;
      od;
      if i mod 10 = 0 then   print(i); fi;
    od;
    print('All tests passed');
  end:
```

This may take a while

```
> TestDeflated(10,10,0.5,0.47, 0.2, [0.0,1.0] );
```

0

All tests passed

Special case: k = 3

All we have to do is to run Marden-Jury test for the deflated characteristic polynomial for k = 3

```
> mu3 := fsolve( subs( c = cos(2*Pi/3), CharPolyMainFactor ), lambda, 0.3..1);
```

μ3 := .4043625706

Verify precision:

```
> inapply( subs( c = cos(2*Pi/3), CharPolyMainFactor ), lambda)(mu3 + 1e-7);
```

$[\,.1029999999\ 10^{-8}, .1170000001\ 10^{-8}\,]$

```
> inapply( subs( c = cos(2*Pi/3),CharPolyMainFactor ), lambda)(mu3 - 1e-7);
```

$[\,-.1180000001\ 10^{-8}, -.1029999999\ 10^{-8}\,]$

```
> intervmu3 := [mu3 - 1e-7, mu3 + 1e-7];
```

intervmu3 := [.4043624706, .4043626706]

```
>
```

Construct the interval deflated polynomial

```
for i from 0 to 2 do
  cf3[i] := inapply(  coeff(subs( c = cos(2*Pi/3), deflate(CharPolyMainFactor/64^5 ,lambda,mu)),lambda,i),  mu);
od:
cf3[3] := mu -> [1.0,1.0]:
deflatedinterv3 := 0:
for i from 0 to 3 do
    deflatedinterv3:= deflatedinterv3 + eval(cf3[i](intervmu3))*lambda^i;
od:
```

Verify that all other roots have magnitude < 0.3: the Marden Jury test for radius 0.3 passes:

```
> IntervMardenJury(deflatedinterv3, lambda, 0.3);
```

table([

1 = [-1.000000005, -.9999999977]

2 = [.9999999952, 1.000000012]

3 = [.9999999902, 1.000000026]

])

Calculation of the largest eigenvalues with guaranteed precision

This function produces a table of approximate values of the eigenvalue with given precision for use with interval arithmetics in the C part of the analysis code; to avoid conversion problems, we write intervals as 3 long integers: low, high, scale, which represent the interval [low/scale, high/scale].

The first value is the limit value for infinity (computed with c set to 1 in the char. polynomial). The result is a C function written to a file; if the file name is 'default', then the output is written to the standard output.

The argument of the function is valence, the function returns the interval value for the largest eigenvalue. The body just contains a big static array.

```
ComputeEigenvalues := proc(N::integer, eps::numeric, fname)
  local K, intervCharPoly,  intervPi, intervc,
  expandedCharPoly, approxEV, r, deflatedCharPoly, i,marTable, cK;
  global SimpleChaprPolyScaled;
  Digits := 15;
  intervCharPoly := inapply( CharPolyMainFactor ,lambda, c);
  # use arccos to make sure we get an interval for Pi
  intervPi := Interval_times([2.0,2.0],Interval_arccos([0,0]));
  intervc := K -> Interval_cos(Interval_times( [2*intervPi[1], 2*intervPi[2]], Interval_reciprocal(map(evalf,[K,K]))));
  fprintf(fname, 'virtual Float Eigenvalue(int K) {\n');
  fprintf(fname, ' static INTEGER64 EV[] = {\n' );
  # do infinty
  expandedCharPoly := subs( c = 1, CharPolyMainFactor);
  approxEV := fsolve( expandedCharPoly, lambda, lambda=0.5..1);
  if op(2, intervCharPoly(approxEV-eps, 1)) > 0
    or op(1,intervCharPoly(approxEV+eps, 1)) < 0  then
   ERROR('fsolve precision failure for infinity');
  fi;
  fprintf( fname, 'CONST64(%d),CONST64(%d),CONST64(%d),\n', op(1,approxEV-eps), op(1,approxEV+eps),10^(-op(2,approxEV)));
  # skip 1,2
  fprintf( fname, 'CONST64(0),CONST64(0),CONST64(0), CONST64(0),CONST64(0),CONST64(0),\n');
  for K from 3 to N  do
    if  (K - 3) mod 100 = 0 then print(K);
    fi;
    cK := intervc(K);
    expandedCharPoly := subs( c = cos(2*Pi/K), CharPolyMainFactor);
    approxEV := fsolve( expandedCharPoly, lambda, lambda=0.25..1);
    # check that the precision is at least eps
    if  op(2, intervCharPoly(approxEV-eps, cK)) > 0
       or op(1, intervCharPoly(approxEV+eps, cK)) < 0  then
      ERROR('fsolve precision failure for K =', K);
    fi;
    # writing out f.p. numbers means relying on the parser of
    # the compiler, which will probably do rounding incorrectly when converting to the
    # binary format; instead, we write out integers and assume that the interval numbers are
    # intialized correctly from integers, i.e. conversion is done by the processor in the correct
    # rounding mode;
    fprintf(fname, 'CONST64(%d),CONST64(%d),CONST64(%d),\n', op(1,eval(approxEV-eps)),
op(1,eval(approxEV+eps)),10^(-op(2,approxEV)));
     od;
    fprintf(fname, 'CONST64(0)};\n return Float(EV[3*K],EV[3*K+1])/Float(EV[3*K+2]) ;\n}\n\n');
  NULL;
end:
```

The derivative of the largest eigenvalue with respect to c at infinity.

To establish C1-continuity for all valences, we need to analyze behavior of the magnitude of the largest eigenvalue as the function of the valence, as the valence increases to infinity (approaches 1). We estimate a constant R, such that $|\lambda - \lambda_\infty| < R|c-1|$, sufficiently close to 1. This constant can be taken to be the maximum of $\left|\frac{\partial}{\partial c}\lambda\right|$, or, equivalently, as maximum of $\left|\frac{\partial}{\partial \lambda} c^{(-1)}\right|$; as the characteristic polynomial is quadratic in c, the latter is relatively easy to compute. Once B is known, we can estimate the size ε of the interval for λ near λ_∞, such that if the characteristic map is injective and regular for all these values, it is sufficient to establish C1-continuity for $K_0 < K$, where $\frac{\varepsilon}{R} < 1 - \cos\left(\frac{2\pi}{K_0}\right)$.

```
> lambdac := [];
  intervlambdac := inapply( -Fc/Flambda, [c,lambda]):
```

lambdac := [ ]

This shows that the maximum of the derivative can be estimated by 0.5:

```
>   for i from 0 to 19 do
      for j from 0 to 9 do
        cinterv := [0.9+i*0.005, 0.905+i*0.005];
        lambdac := Interval_union(lambdac,intervlambdac( cinterv, [0.47+0.2*cinterv[1]+j*0.005, 0.48+0.2*cinterv[2]+j*0.005]));
      od;
    od; eval(lambdac);
```

[.09504546976, .4063409907]

```
> R := op(2, lambdac);
```

R := .4063409907

```
>
```

Code generation