1D Model Problem

Construct the typical tridiagonal Matrix for the 1D Poisson equation with zero right hand side vector b:

```maple
restart;
with(LinearAlgebra):
with(plots):

1D Model Problem

Construct the typical tridiagonal Matrix for the 1D Poisson equation with zero right hand side vector b:

> n := 7;
> A := Matrix(
[ seq(-1, i=1..n-1) ],
[ seq(2, i=1..n) ],
[ seq(-1, i=1..n-1) ], shape=band[1,1],
scan=band[1,2]);

> b := <seq(0, i=1..n)>;

> evalf(Eigenvalues(A));

Conjugate Gradients

Standard CG-algorithm without preconditioning:

> CG := proc(A::Matrix, b::Vector, xstart::Vector, it::posint)
#option trace;
local i, alpha, beta, x, r, resnorm, d:
x[0] := xstart;
r := b - A.x[0];
d := r;
```
for i from 1 to it do
    resnorm := DotProduct(r, r);
    alpha := resnorm / BilinearForm(d, d, A);
    x[i] := x[i-1] + alpha * d;
    r := r - alpha * A . d;
    beta := DotProduct(r, r) / resnorm;
    d := r + beta * d;
end do;
return x;
end proc;

CG := proc(A::Matrix, b::Vector, xstart::Vector, it::posint)
local i, alpha, beta, x, r, resnorm, d;
x[0] := xstart;
r := b - (A . x[0]);
d := r;
for i to it do
    resnorm := LinearAlgebra:-DotProduct(r, r);
    alpha := resnorm / LinearAlgebra:-BilinearForm(d, d, A);
    x[i] := x[i-1] + alpha * d;
    r := r - ((alpha*A) . d);
    beta := LinearAlgebra:-DotProduct(r, r) / resnorm;
    d := r + beta * d
end do;
return x
end proc;

> xstart := evalf(RandomVector(n));

# try an eigenvector instead:
# k := 5;
# xstart := < seq( evalf(sin(k*Pi*j/(n+1))), j=1..n) >;

> iter := 20;
> xs := CG(A, b, xstart, iter);

> listplot([ 0, seq(xs[6][i], i=1..n), 0]);
plot the maximum errors of each iterative solution:

```maple
> plotlist := [ seq( listplot([ 0, seq(xs[j][i], i=1..n), 0]),
                   j=0..iter ) ]:
display(plotlist, insequence=true);
```

```maple
> listplot( [ seq( Norm(xs[j]), j=0..iter ) ]);
```
Preconditioned Conjugate Gradient - Hierarchical Basis Preconditioning

Hierarchical Basis Transformation

hierarch corresponds to the inverse of the matrix L of the lecture slides (transforms coefficients for nodal basis into coefficients for hierarchical basis)

```maple
> hierarch := proc( x::Vector, n::posint )
# option trace;
local l,i, left, right, xx;
xx := evalf(x);
l := 1;
while l < n/2 do
    for i from l to n by 2*l do
        if i-l>0 then left := xx[i-l] else left:=0 end if;
        if i+l<=n then right := xx[i+l] else right:=0 end if;
        xx[i] := xx[i]-(left+right)/2;
    end do;
l := l*2;
end do;
return xx;
end proc:
```

dehierarch corresponds to matrix L on the lecture slides (transforms coefficients for hierarchical basis into coefficients for nodal basis)

```maple
> dehierarch := proc( x::Vector, n::posint )
```

---

Graph with data points and a downward trend starting from the top left corner, showing a decrease as it moves towards the bottom right corner.
dehierarchT corresponds to the transpose of matrix L
(applied to a vector of nodal basis function, it will transform this into a vector of
hierarchical basis functions)

> dehierarchT := proc( x::Vector, n::posint )
  #option trace;
  local l,i,left,right,xx;
  xx := evalf(x);
  l := 1;
  while l <= (n+1)/4 do
    for i from 2*l to n by 2*l do
      xx[i] := xx[i]+(xx[i-l]+xx[i+l])/2;
    end do;
    l := l*2;
  end do;
  return xx;
end proc:

Example for hierarchization - use a sin-mode as example:
> k := 1;
xstart := < seq( evalf(sin(k*Pi*j/(n+1))), j=1..n ) >;
k := 1

xstart :=
  0.3826834325
  0.7071067810
  0.9238795325
  1.
  0.9238795325
  0.7071067810
  0.3826834325

> hierarch(xstart,n);
CG with Hierarchical-Basis Preconditioning

```plaintext
> PCG := proc(n::posint, A::Matrix, b::Vector, xstart::Vector, it::posint)
  local i, alpha, beta, x, r, resnorm, d, Ad:
  x[0] := xstart;
  r := dehierarchT(b - A.x[0], n);
  d := dehierarch(r, n);
  for i from 1 to it do
    resnorm := DotProduct(r, r);
    Ad := A.d;
    alpha := resnorm / DotProduct(d, Ad);
    x[i] := x[i-1] + alpha * d;
    r := r - alpha * dehierarchT(Ad, n);
    beta := DotProduct(r, r) / resnorm;
    d := dehierarch(r, n) + beta * d;
  end do;
  return x;
end proc:

> n := 7;
A := Matrix( [ [ seq(-1, i=1..n-1) ],
               [ seq(2, i=1..n) ],
               [ seq(-1, i=1..n-1) ] ], shape=band[1,1],
 scan=band[1,2]);
b := < seq(0, i=1..n) >;
xstart := evalf( RandomVector(n) );
# k := 3;
# xstart := < seq( evalf(sin(k*Pi*j/(n+1))), j=1..n) >;
n := 7
```
Plot the solution after the first iteration $\rightarrow$ basically $=0$ (hence, convergence in a single step).
**Explanation: Compute the Preconditioning Matrices**

The matrices $R$ (for the hierarchical transform) and $L$ (for de-hierarchization) can be computed column-wise by applying the transforms to all unit vectors. Note that in Maple notation $<a,b,c>$ denotes a matrix, if $a$, $b$, and $c$ are row vectors of the matrix.

```maple
> n := 7;
A := Matrix( [ [ seq(-1, i=1..n-1) ],
               [ seq(2, i=1..n) ],
               [ seq(-1, i=1..n-1) ] ], shape=band[1,1],
            scan=band[1,2]);

> R := Transpose( < seq( Transpose( hierarch( < seq("if"(i=j,1,0),j=1..n) >,n) ), i=1..n ) > );

> L := Transpose( < seq( Transpose( dehierarch( < seq("if"(i=j,1,0),j=1..n) >,n) ), i=1..n ) > );
```
Note that $R$ and $L$ are inverse to each other (as hierarchical "forward" and "backward" transformations)

```
> R . L;
```

Most important: $L^T A L$ is a diagonal matrix (which explains the rapid CG convergence); note that the size of the diagonal levels depends on the hierarchical level of the corresponding basis functions!

```
> LAL := Transpose(L) . Matrix(A) . L;
```

Apply regular CG algorithm to the transformed system of equations;
note that we transform the starting solution $x_{start}$ into hierarchical-basis form:

```maple
> iter := 3;
xs := CG(LAL, dehierarchT(b,n), hierarch(xstart,n), iter);
solplot := [ seq( listplot([ 0, seq(xs[j][i], i=1..n), 0]), j=0..iter ) ];
display(solplot, insequence=true);
```

```maple
iter := 3
xs := x
```