restart;
gaussseidel := proc(A::Matrix,b::Vector,x0::Vector,tol::positive,
n::nonnegint,v::name)
local AA, B, OK, N, I, J, X0, X1, TOL, NN, K, ERR, S, X;
with(LinearAlgebra);
N:=RowDimension(A);
AA:=A;
B:=b;
X0:=x0;
TOL:=tol;
NN:=n;
X1:=Vector(N);
STEP 1
> K := 0;
> OK := FALSE;
printf(` n x\n`);
printf(` - -\n`);
printf(`%3d [%`);K);
for I from 1 to N do
printf(`% 12.8f`,X0[I]);
if I<=N then printf(` ,`) fi;
od;
printf(`]\n`);
K:=1;
STEP 2
> while OK = FALSE and K <= NN do
err is used to test accuracy - it measures the infinity-norm
> ERR := 0;
STEP 3
> for I from 1 to N do
> S := 0;
> for J from 1 to I-1 do
> S := S-A[I,J]*X1[J];
> od;
> for J from I+1 to N do
> S := S-A[I,J]*X0[J];
> od;
> S := evalf((S+B[I])/A[I,I]);
if abs(S-X0[I]) > ERR then
ERR := abs(S-X0[I]);
fi;
use X1 for X
> X1[I] := S;
> od;
printf(`%3d [%`);K);
for I from 1 to N do
printf(`% 12.8f`,X1[I]);
if I<=N then printf(` ,`) fi;
od;
printf(`]\n`);
STEP 4
> if ERR <= TOL then
> OK := TRUE;
> fi;
process is complete
STEP 5
> K := K+1;
STEP 6
> for I from 1 to N do
> X0[I] := X1[I];
> od;
> od;
> if OK = FALSE then
> printf(`Maximum Number of Iterations Exceeded.`);
> end if;
> printf(`% 12.8f`,
X0 
[I 
]);
> if I < N then
> printf (`,
end if
end do;
printf (`
`);
K := 1;

STEP 7

procedure completed unsuccessfully

> else
> printf(`The solution vector is

`,args[6]);
> v:=evalm(X1);
f;
fi;
end;

Warning, imaginary unit `I` used as a local variable in procedure gaussseidel
gaussseidel := proc(A::Matrix, b::Vector, x0::Vector, tol::positive, n::nonnegint, v::name)

  local AA, B,

  OK, N, I, J, X0, X1, TOL, NN, K, ERR, S, X;

  with(LinearAlgebra);

  N := RowDimension(A);
  AA := A;
  B := b;
  X0 := x0;
  TOL := tol;
  NN := n;
  X1 := Vector(N);
  K := 0;

  OK := FALSE;

  printf(`

`);

  printf(` - -

`);

  printf(`%3d [`,K);

  for I to N do

    printf(`% 12.8f`,
X0[I]);

    if I <> N then

      printf(``
end if

end do;

printf(`
`);

K := 1;
while OK = FALSE and K <= NN do

ERR := 0;
for I to N do
  S := 0;
  for J to I - 1 do
  end do;
  for J from I + 1 to N do
    S := S - A[I, J] * X0[J]
  end do;
  S := evalf((S + B[I]) / A[I, I]);
  if ERR < abs(S - X0[I]) then
    ERR := abs(S - X0[I])
  end if;
  X1[I] := S
end do;
printf(`%3d `, K);
for I to N do
  printf(`%12.8f `, X1[I]);
  if I <> N then
    printf(``, `)
  end if
end do;
printf(`
`);
if ERR <= TOL then
  OK := TRUE
end if;
K := K + 1;
for I to N do
  X0[I] := X1[I]
end do
end do;
if OK = FALSE then
  printf(`Maximum Number of Iterations Exceeded.
`
else

printf(`
    The solution vector is
`, args[6]);

v := evalm(X1)
end if
end proc

> gaussseidel_dir:=proc()
    printf(`gaussseidel returns an approximation to a solution of a
    vector equation.\n\n`);
    printf(`The arguments for gauss-seidel are:\n`);
    printf(`(1)the coefficient matrix (must be square)\n`);
    printf(`(2)the right hand side vector\n`);
    printf(`(3)the initial approximation vector\n`);
    printf(`(4)tolerance\n`);
    printf(`(5)maximum number of iterations\n`);
    printf(`(6)variable for returning the approximate solution\n`);
    printf(`If assigning the result to a variable, have the\n`);
    printf(`variable and the 6th argument the same.\n`);
    printf(`If v is the variable for returning the approximate\n    solution\n`);
    printf(`and has already been given a value,\n`);
    printf(`the procedure should be preceded by the statement:\n`);
    printf(`v:=`v`\n`);
end;

gaussseidel_dir := proc ( )

    printf(
        `gaussseidel returns an approximation to a solution of a vector equation.
        `);

    printf(`The arguments for gauss-seidel are:
        `);
    printf(`(1)the coefficient matrix (must be square)
        `);
    printf(`(2)the right hand side vector
        `);
    printf(`(3)the initial approximation vector
        `);
    printf(`(4)tolerance
        `);
    printf(`(5)maximum number of iterations
        `);
    printf(`(6)variable for returning the approximate solution
        `);

`\);
printf(`If assigning the result to a variable, have the
`);
printf(`variable and the 6th argument the same.
`);
printf(`If v is the variable for returning the approximate solution
`);
printf(`and has already been given a value,
`);
printf(`the procedure should be preceded by the statement:
`);
printf(`v:='v``)
end proc