Freie Randbedingungen (N1 = N2 = 7).

```
> restart:

  with(linalg):
  with(plots):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
Warning, the name changecoords has been redefined
```

```
>
  N1 := 7;  # Anzahl der Massenpunkte in 1-Richtung.
  N2 := 7;  # Anzahl der Massenpunkte in 2-Richtung.

  m := 1.0;  # Masse.
  k := 1.0;  # Federkonstante.


  # Massenmatrix.

  M := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do
    for index2 from 1 to N1*N2 do

      M[index1,index2] := 0:

    od:
  od:

  for i1 from 1 to N1 do
    for i2 from 1 to N2 do

      index_ := (i2-1)*N1 + i1:

      M[index_,index_] := m:

    od:
  od:

  # evalm(M);
```

```
# Kraftmatrix (Aufgabe 1.(a) --> freie Randbedingungen).

K := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do
  for index2 from 1 to N1*N2 do

    K[index1,index2] := 0:

  od:
od:

for i1 from 1 to N1 do
  for i2 from 1 to N2 do

    index_ := (i2-1)*N1 + i1:

    if (i1 <> N1) then

      # Massenpunkt befindet sich nicht am rechten Rand.

      index__ := (i2-1)*N1 + i1+1:

      K[index_,index_] := K[index_,index_] + k:
      K[index__,index__] := K[index__,index__] + k:
      K[index_,index__] := K[index_,index__] - k:
      K[index__,index_] := K[index__,index_] - k:

    fi:

    if (i2 <> N2) then

      # Massenpunkt befindet sich nicht am oberen Rand.

      index__ := i2*N1 + i1:

      K[index_,index_] := K[index_,index_] + k:
      K[index__,index__] := K[index__,index__] + k:
      K[index_,index__] := K[index_,index__] - k:
      K[index__,index_] := K[index__,index_] - k:

    fi:

  od:
od:
```

```
  # evalm(K);
```

$$N1 := 7$$
$$N2 := 7$$
$$m := 1.0$$
$$k := 1.0$$

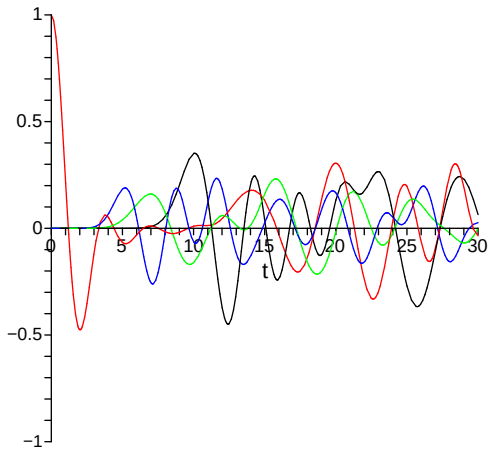
```
>
  # Eigenfrequenzen und Eigenvektoren berechnen.
  result := [eigenvectors(multiply(inverse(M), K))]:
```

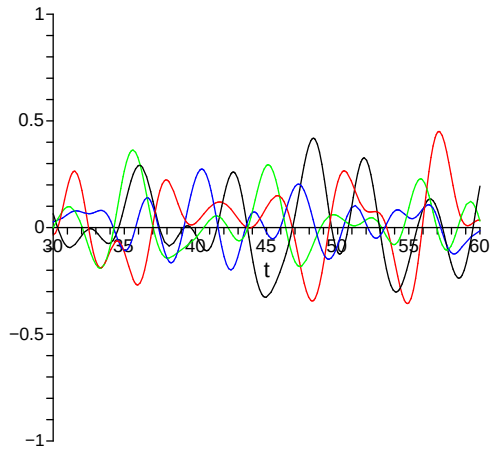
```
>
  # Eigenfrequenzen und Eigenvektoren zur Loesung des Anfangswertproblem
  vorbereiten.

  ctr := 0:

  # Die Eigenfrequenzen.
  omega := vector(N1*N2):

  # Die Spalten dieser Matrix entsprechen den Eigenvektoren.
  v := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do

    for j1 from 1 to result[index1][2] do

      ctr := ctr+1:

      omega[ctr] := sqrt(abs(result[index1][1])):

      for index2 from 1 to N1*N2 do
        v[index2,ctr] := result[index1][3][j1][index2]:
      od:

    od:

    if (ctr = N1*N2) then
      break:
    fi:

  od:

  evalm(omega);
  # evalm(v);
```

```
# Anfangswertproblem loesen.


# Anfangspositionen.

x_0 := vector(N1*N2):

for index1 from 1 to N1*N2 do
  x_0[index1] := 0:
od:

x_0[1] := 1:

# evalm(x_0);

a := multiply(inverse(v), x_0):

# evalm(a);


# Anfangsgeschwindigkeiten --> trivial.


# Endergebnis.

for index1 from 1 to N1*N2 do

  a[index1] := a[index1] * cos(omega[index1]*t):

od:

x := multiply(v, a):

# evalm(x);
# simplify(subs(t=0, evalm(x)));
```

```
[0.0001414213562, 2.499395851, 1.563662966, 2.314496894, 1.801937737, 1.801937737,
    0.9752346703, 1.519203254, 1.246979604, 2.757512551, 0.9752346695, 2.191332414,
    0.8677674819, 2.000000002, 1.856082399, 1.949855826, 2.191332412, 2.314496893,
    2.548324787, 1.519203256, 0.4450418687, 1.788312688, 2.385795775, 2.385795775,
    1.227208538, 1.625762632, 2.499395849, 2.654979726, 1.324016767, 2.134234788,
    1.788312687, 2.134234791, 1.324016768, 1.625762632, 2.211353375, 2.654979727,
    2.000000000, 2.000000000, 2.000000000, 1.563662967, 0.8677674824, 1.763495468,
    1.949855825, 0.4450418520, 1.246979605, 0.6293842516, 1.999999999, 2.000000000
    1.856082398]
```

```
> # Teilchen 1, 7, 25, 49.
  plot([x[1], x[7], x[25], x[49]], t=00.0..30.0,
  view=[00.0..30.0,-1.0..1.0], color=[red,green,blue,black]);
  plot([x[1], x[7], x[25], x[49]], t=30.0..60.0,
  view=[30.0..60.0,-1.0..1.0], color=[red,green,blue,black]);
```






```
>
```

Freie Randbedingungen, mittlere Zeile schwach gekoppelt (N1 = N2 = 7).

```
> restart:

  with(linalg):
  with(plots):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
Warning, the name changecoords has been redefined
```

```
>
  N1 := 7;  # Anzahl der Massenpunkte in 1-Richtung.
  N2 := 7;  # Anzahl der Massenpunkte in 2-Richtung.

  m := 1.0;  # Masse.
  k := 1.0;  # Federkonstante.


  # Massenmatrix.

  M := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do
    for index2 from 1 to N1*N2 do

      M[index1,index2] := 0:

    od:
  od:

  for i1 from 1 to N1 do
    for i2 from 1 to N2 do

      index_ := (i2-1)*N1 + i1:

      M[index_,index_] := m:

    od:
  od:

  # evalm(M);


  # Kraftmatrix (Aufgabe 1.(a) --> freie Randbedingungen).

  K := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do
    for index2 from 1 to N1*N2 do

      K[index1,index2] := 0:

    od:
  od:

  for i1 from 1 to N1 do
    for i2 from 1 to N2 do

      index_ := (i2-1)*N1 + i1:

      if (i1 <> N1) then

        # Massenpunkt befindet sich nicht am rechten Rand.

        index__ := (i2-1)*N1 + i1+1:

        if (i2 = 4) then
          K[index_,index_] := K[index_,index_] + k/5:
          K[index__,index__] := K[index__,index__] + k/5:
          K[index_,index__] := K[index_,index__] - k/5:
          K[index__,index_] := K[index__,index_] - k/5:
        else
          K[index_,index_] := K[index_,index_] + k:
          K[index__,index__] := K[index__,index__] + k:
          K[index_,index__] := K[index_,index__] - k:
          K[index__,index_] := K[index__,index_] - k:
        fi:

      fi:

      if (i2 <> N2) then

        # Massenpunkt befindet sich nicht am oberen Rand.

        index__ := i2*N1 + i1:

        if (i2 = 3 or i2 = 4) then
          K[index_,index_] := K[index_,index_] + k/5:
          K[index__,index__] := K[index__,index__] + k/5:
          K[index_,index__] := K[index_,index__] - k/5:
          K[index__,index_] := K[index__,index_] - k/5:
        else
          K[index_,index_] := K[index_,index_] + k:
          K[index__,index__] := K[index__,index__] + k:
          K[index_,index__] := K[index_,index__] - k:
          K[index__,index_] := K[index__,index_] - k:
        fi:

      fi:

    od:
  od:

  # evalm(K);
```

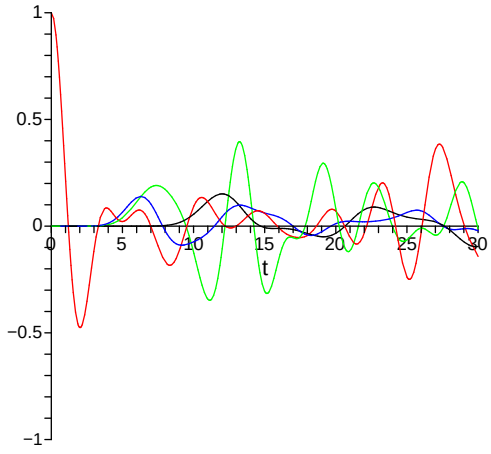
$$N1 := 7$$
$$N2 := 7$$
$$m := 1.0$$
$$k := 1.0$$

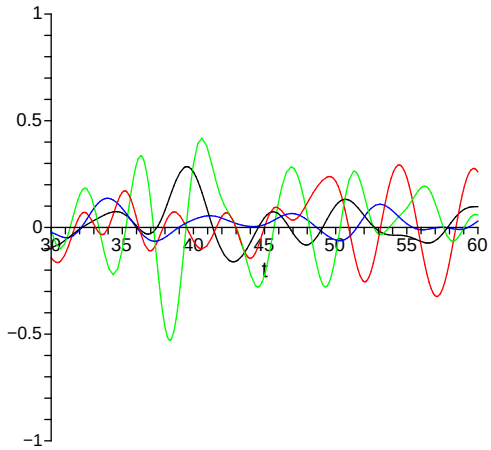
```
> # Eigenfrequenzen und Eigenvektoren berechnen.
  result := [eigenvectors(multiply(inverse(M), K))]:

> # Eigenfrequenzen und Eigenvektoren zur Loesung des Anfangswertproblems
  vorbereiten.

  ctr := 0:

  # Die Eigenfrequenzen.
  omega := vector(N1*N2):

  # Die Spalten dieser Matrix entsprechen den Eigenvektoren.
  v := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do

    for j1 from 1 to result[index1][2] do

      ctr := ctr+1:

      omega[ctr] := sqrt(abs(result[index1][1])):

      for index2 from 1 to N1*N2 do
        v[index2,ctr] := result[index1][3][j1][index2]:
      od:

    od:

    if (ctr = N1*N2) then
      break:
    fi:

  od:

  evalm(omega);
  # evalm(v);
```

```
  # Anfangswertproblem loesen.


  # Anfangspositionen.

  x_0 := vector(N1*N2):

  for index1 from 1 to N1*N2 do
    x_0[index1] := 0:
  od:

  x_0[1] := 1:

  # evalm(x_0);

  a := multiply(inverse(v), x_0):

  # evalm(a);


  # Anfangsgeschwindigkeiten --> trivial.


  # Endergebnis.

  for index1 from 1 to N1*N2 do

    a[index1] := a[index1] * cos(omega[index1]*t):

  od:

  x := multiply(v, a):

  # evalm(x);
  # simplify(subs(t=0, evalm(x)));
```

[1.884014486, 2.614922872, 1.630774922, 0.5076511149, 1.162199090, 1.362897321, 0.00008816475713, 2.086012538, 2.615445059, 1.888327010, 1.967049306, 1.050937065, 1.820933001, 2.341137707, 1.279221748, 2.341859507, 1.066950840, 1.798317593, 1.744073328, 2.506564084, 2.217652096, 1.078045154, 2.143559221, 2.215040959, 1.141284793, 0.8142802236, 1.270671286, 2.507155259, 1.582621008, 0.9014808817, 0.6293066555, 0.9255801904, 1.374821707, 1.947744017, 2.089155999, 0.6750044175, 1.586616035, 1.637546740, 1.818413317, 1.742378804, 1.799865795, 0.6912486989, 1.946510801, 1.965091632, 0.4071251493, 0.9328412638, 0.2442281636, 2.142625033, 1.011996766]

```
> # Teilchen 1, 7, 25, 49.

  plot([x[1], x[7], x[25], x[49]], t=00.0..30.0,
  view=[00.0..30.0,-1.0..1.0], color=[red,green,blue,black]);
  plot([x[1], x[7], x[25], x[49]], t=30.0..60.0,
  view=[30.0..60.0,-1.0..1.0], color=[red,green,blue,black]);
```






```
>
```

Freie Randbedingungen, zentraler Massenpunkt an die Umgebung gekoppelt (N1 = N2 = 7).

```
> restart:

  with(linalg):
  with(plots):

Warning, the protected names norm and trace have been redefined and
unprotected
Warning, the name changecoords has been redefined
```

```
>
N1 := 7;  # Anzahl der Massenpunkte in 1-Richtung.
N2 := 7;  # Anzahl der Massenpunkte in 2-Richtung.

m := 1.0;  # Masse.
k := 1.0;  # Federkonstante.


# Massenmatrix.

M := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do
  for index2 from 1 to N1*N2 do

    M[index1,index2] := 0:

  od:
od:

for i1 from 1 to N1 do
  for i2 from 1 to N2 do

    index_ := (i2-1)*N1 + i1:

    M[index_,index_] := m:

  od:
od:

# evalm(M);


# Kraftmatrix (Aufgabe 1.(a) --> freie Randbedingungen).

K := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do
  for index2 from 1 to N1*N2 do

    K[index1,index2] := 0:

  od:
od:

for i1 from 1 to N1 do
  for i2 from 1 to N2 do

    index_ := (i2-1)*N1 + i1:

    if (i1 <> N1) then

      # Massenpunkt befindet sich nicht am rechten Rand.

      index__ := (i2-1)*N1 + i1+1:

      K[index_,index_] := K[index_,index_] + k:
      K[index__,index__] := K[index__,index__] + k:
      K[index_,index__] := K[index_,index__] - k:
      K[index__,index_] := K[index__,index_] - k:

    fi:

    if (i2 <> N2) then

      # Massenpunkt befindet sich nicht am oberen Rand.

      index__ := i2*N1 + i1:

      K[index_,index_] := K[index_,index_] + k:
      K[index__,index__] := K[index__,index__] + k:
      K[index_,index__] := K[index_,index__] - k:
      K[index__,index_] := K[index__,index_] - k:

    fi:

    if (i1 = 4 and i2 = 4) then
      K[index_,index_] := K[index_,index_] + 5*k:
    fi:

  od:
od:

# evalm(K);
```

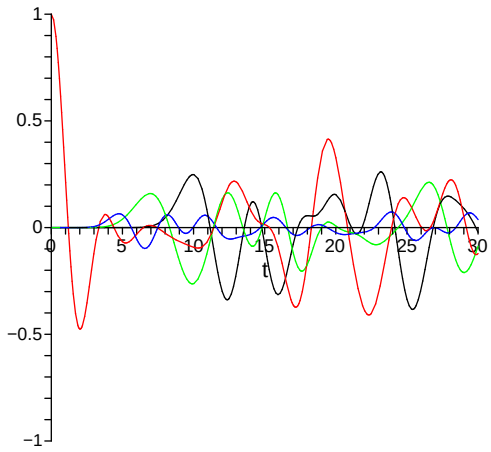
$$N1 := 7$$
$$N2 := 7$$
$$m := 1.0$$
$$k := 1.0$$

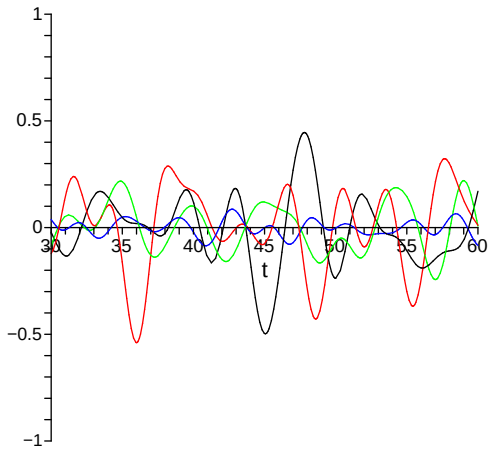
```
>
# Eigenfrequenzen und Eigenvektoren berechnen.
result := [eigenvectors(multiply(inverse(M), K))]:
```

```
>
# Eigenfrequenzen und Eigenvektoren zur Loesung des Anfangswertproblems
vorbereiten.

ctr := 0:

# Die Eigenfrequenzen.
omega := vector(N1*N2):

# Die Spalten dieser Matrix entsprechen den Eigenvektoren.
v := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do

  for j1 from 1 to result[index1][2] do

    ctr := ctr+1:

    omega[ctr] := sqrt(abs(result[index1][1])):

    for index2 from 1 to N1*N2 do
      v[index2,ctr] := result[index1][3][j1][index2]:
    od:

  od:

  if (ctr = N1*N2) then
    break:
  fi:

od:

evalm(omega);
# evalm(v);


# Anfangswertproblem loesen.


# Anfangspositionen.

x_0 := vector(N1*N2):

for index1 from 1 to N1*N2 do
  x_0[index1] := 0:
od:

x_0[1] := 1:

# evalm(x_0);

a := multiply(inverse(v), x_0):

# evalm(a);


# Anfangsgeschwindigkeiten --> trivial.


# Endergebnis.

for index1 from 1 to N1*N2 do

  a[index1] := a[index1] * cos(omega[index1]*t):

od:

x := multiply(v, a):

# evalm(x);
# simplify(subs(t=0, evalm(x)));
```

```
[1.999999997, 1.246979606, 2.654979725, 0.6293842485, 0.1897857845, 2.385795773,
    1.519203254, 1.801937736, 2.000000001, 2.191332411, 0.9752346654, 1.246979605
    2.000000000, 1.856082397, 1.788312686, 1.763495468, 0.4450418683, 1.949855824
    1.294305280, 1.999999996, 1.324016768, 1.607719274, 0.9752346722, 1.856082398
    2.677569660, 1.563662965, 0.4450418674, 2.355049780, 2.385795775, 1.864881702
    2.191332410, 2.314496891, 1.519203255, 0.9466274986, 2.548324784, 1.625762631
    1.999999999, 2.184345995, 2.499395847, 2.004291238, 0.8677674767, 1.324016766
    1.625762630, 2.314496891, 2.000000000, 1.801937735, 2.134234788, 2.654979725,
    3.134631877]
```

```
>
# Teilchen 1, 7, 25, 49.
plot([x[1], x[7], x[25], x[49]], t=00.0..30.0,
view=[00.0..30.0,-1.0..1.0], color=[red,green,blue,black]);
plot([x[1], x[7], x[25], x[49]], t=30.0..60.0,
view=[30.0..60.0,-1.0..1.0], color=[red,green,blue,black]);
```





>

Periodische Randbedingungen (N1 = N2 = 4).

```
>
  restart:

  with(linalg):
  with(plots):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
Warning, the name changecoords has been redefined
```

```
>
```

```
N1 := 4;  # Anzahl der Massenpunkte in 1-Richtung.
N2 := 4;  # Anzahl der Massenpunkte in 2-Richtung.

m := 1;  # Masse.
k := 1;  # Federkonstante.


# Massenmatrix.

M := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do
  for index2 from 1 to N1*N2 do

    M[index1,index2] := 0:

  od:
od:

for i1 from 1 to N1 do
  for i2 from 1 to N2 do

    index_ := (i2-1)*N1 + i1:

    M[index_,index_] := m:

  od:
od:

evalm(M);


# Kraftmatrix (Aufgabe 1.(b) --> periodische Randbedingungen).

K := matrix(N1*N2, N1*N2):

for index1 from 1 to N1*N2 do
  for index2 from 1 to N1*N2 do

    K[index1,index2] := 0:

  od:
od:

for i1 from 1 to N1 do
  for i2 from 1 to N2 do

    index_ := (i2-1)*N1 + i1:

    index__ := (i2-1)*N1 + (i1 mod N1)+1:

    K[index_,index_] := K[index_,index_] + k:
    K[index__,index__] := K[index__,index__] + k:
    K[index_,index__] := K[index_,index__] - k:
    K[index__,index_] := K[index__,index_] - k:

    index__ := (i2 mod N2)*N1 + i1:

    K[index_,index_] := K[index_,index_] + k:
    K[index__,index__] := K[index__,index__] + k:
    K[index_,index__] := K[index_,index__] - k:
    K[index__,index_] := K[index__,index_] - k:

  od:
od:

evalm(K);
```

$$N1 := 4$$

$$N2 := 4$$

$$m := 1$$

$$k := 1$$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

$$\begin{bmatrix}
4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
-1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4
\end{bmatrix}$$

```
                    [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, -1, 4, -1],
                    [0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, -1, 4]]
```

```
> # Eigenfrequenzen und Eigenvektoren berechnen.
  result := [eigenvectors(multiply(inverse(M), K))]:
```

```
> # Eigenfrequenzen und Eigenvektoren zur Loesung des Anfangswertproblems
  vorbereiten.

  ctr := 0:

  # Die Eigenfrequenzen.
  omega := vector(N1*N2):

  # Die Spalten dieser Matrix entsprechen den Eigenvektoren.
  v := matrix(N1*N2, N1*N2):

  for index1 from 1 to N1*N2 do

    for j1 from 1 to result[index1][2] do

      ctr := ctr+1:

      omega[ctr] := sqrt(abs(result[index1][1])):

      for index2 from 1 to N1*N2 do
        v[index2,ctr] := result[index1][3][j1][index2]:
      od:

    od:

    if (ctr = N1*N2) then
      break:
    fi:

  od:

  evalm(omega);
  evalm(v);


  # Anfangswertproblem loesen.


  # Anfangspositionen.

  x_0 := vector(N1*N2):

  for index1 from 1 to N1*N2 do
    x_0[index1] := 0:
  od:

  x_0[1] := 1:

  # evalm(x_0);

  a := multiply(inverse(v), x_0):

  # evalm(a);


  # Anfangsgeschwindigkeiten --> trivial.


  # Endergebnis.

  for index1 from 1 to N1*N2 do

    a[index1] := a[index1] * cos(omega[index1]*t):

  od:

  x := multiply(v, a):

  # evalm(x);
  # simplify(subs(t=0, evalm(x)));
```

$$[\sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, 2, 2, 2, 2, 2, 0, 2\sqrt{2}, \sqrt{6}, \sqrt{6}, \sqrt{6}, \sqrt{6}]$$

```
[[1, 1, -1, -2, 0, 0, 0, 0, 1, 0, 1, -1, 1, 0, 1, 1], [0, 1, -1, -1, 1, 0, 0, 0, 0, 0, 1, 1, -1, 0, 0, 0],
    [-1, 0, 0, 0, 0, -1, 0, -1, -1, 0, 1, -1, 1, -1, -1, 0],
    [0, 0, 0, -1, -1, 0, -1, 0, 0, -1, 1, 1, -1, 1, 0, -1],
    [1, 0, -1, -1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, -1, 0],
    [0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, -1],
    [-1, -1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1],
    [0, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -1, 0, -1, 0, 0],
    [1, 0, 0, 0, 0, 0, -1, 0, -1, -1, 0, 1, -1, -1, 1, 1, 0],
    [0, 0, 0, 0, 1, -1, 0, -1, 0, 0, -1, 1, 1, 1, -1, 0, 1],
    [-1, -1, 1, 2, 0, 0, 0, 0, 1, 0, 1, -1, -1, 0, -1, -1],
    [0, -1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
    [1, 1, 0, -1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, -1, -1, -1],
    [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -1, 0, 1, 0, 0], [-1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0],
    [0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 1]]
```

```
>
```