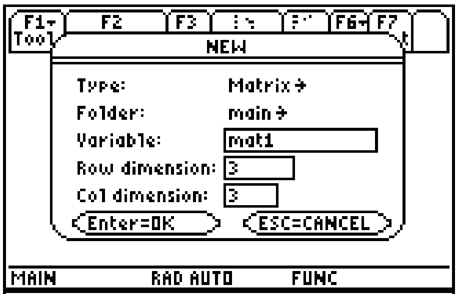
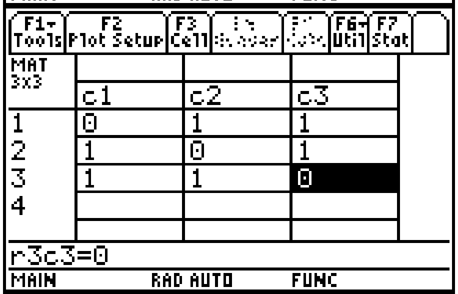
Eigenvalue/Eigenvector finder for quadratic matrices
By Marcel Brätz (marc@mbraetz.de, http://mbraetz.de)

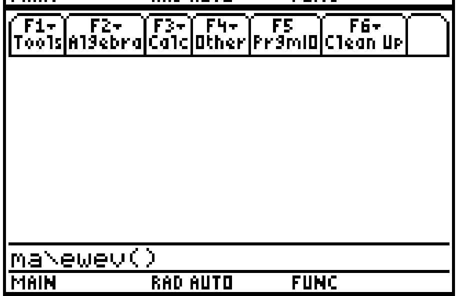
Step by step description for using the program



First you need to create a matrix using the matrix editor of your calculator. The matrix has to be quadratic!!! Otherwise the program will generate an error message.

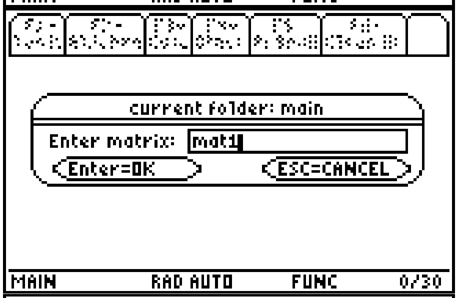


After creating the matrix you can enter the desired values.



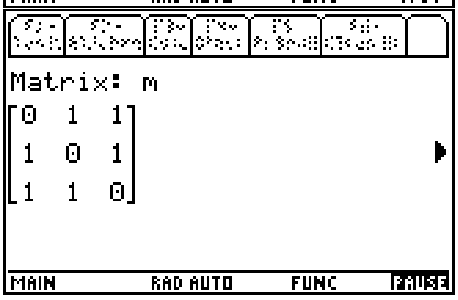
Now, run the program.

There are no parameters required.

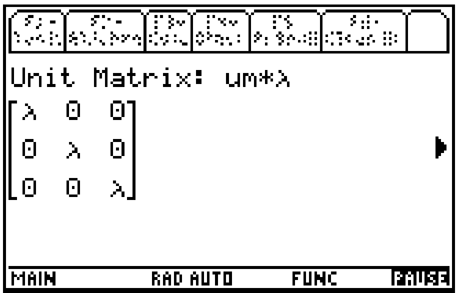


As soon as the program starts, it will ask for the name of a matrix to load.

Note: The current folder is displayed in the Title of the window for orientation.

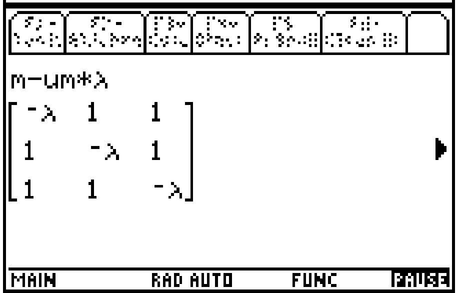


The matrix is then displayed.

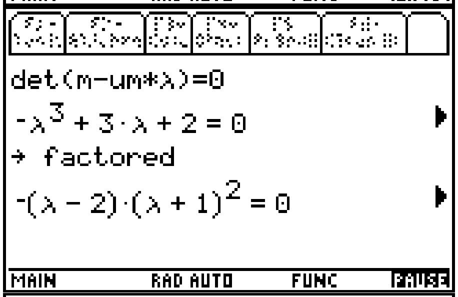


The program now generates a second matrix with the same dimension as the one you entered with 1 in the main diagonal.

That matrix is a unit matrix multiplied by λ.



This unit matrix is now subtracted from the matrix you entered.
(I will call this matrix resulting matrix later in the description.)

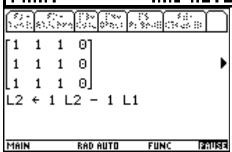


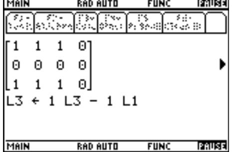
The program now generates the determinant of the resulting matrix.

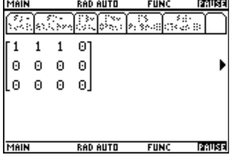
This returns a polynomial equation of the same degree as the row/column number of the matrix you entered.

The next step gives you the Zeros of that equation.
Usually there are just as many as there are rows/columns in the matrix. If there are less, like here, then one of them occurs several times or they are non-real.

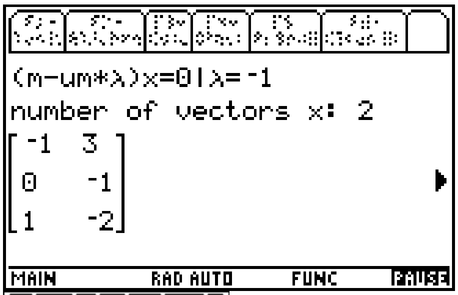
The calculated eigenvalues are now used to solve the vector equation
$(A - E \cdot \lambda)\vec{x} = \vec{0}$
A is your matrix, E is the unit matrix, λ is the current eigenvalue and $\vec{x}$ is the eigenvector to solve for.



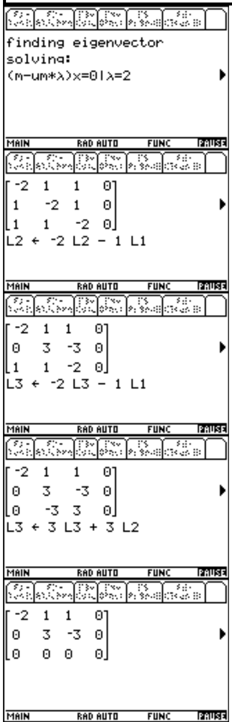




Program now uses the Gauss algorithm to simplify the (n,n+1) that results from the augmentation of the resulting matrix an one column of 0 ($\vec{0}$).
This matrix represents a system of linear equations and can be solved like one.
This is done for every λ.
The number of lines consisting only of zeros is the number of solutions for λ.
Every step is documented so you can write them down in a test.

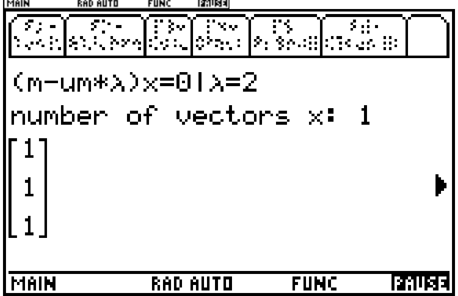


The solution is now displayed.

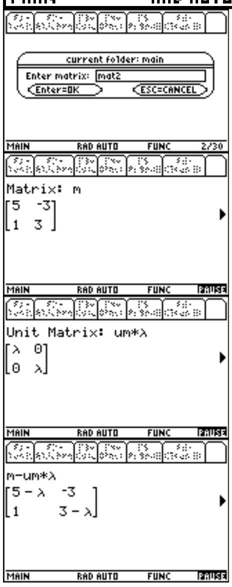


The Gauss algorithm is executed just as many times as there are different eigenvalues. Every time the program returns every step for writing them down in a test.

Note: Don't tell the teacher about this program, cause you might not be able to use the calculator in a test for this.



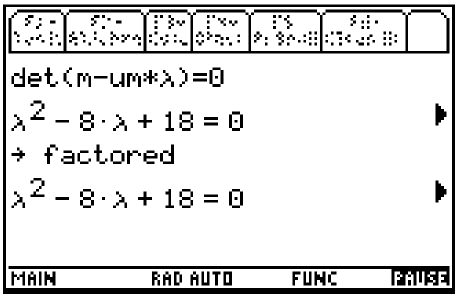
For each eigenvalue λ the eigenvectors are displayed.



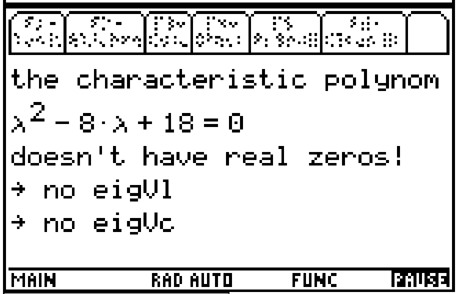
Sometimes a matrix does not produce eigenvalues and therefore no eigenvectors.

The following steps just document what happens when such an event occurs.

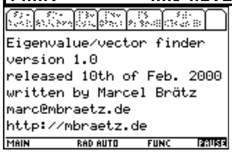
You basically follow the same procedure as described above.



When the factorization of the polynonial equation of the determinant does not produce any linear factors (such as $(x+2)$ or $(x-9)^2$) then the result is non-real.



The program will recognize this situation and give you the proper hint.



The last screen is the copyright info.
You may use any part of the program you like, but do not touch the program itself!!!

The Gauss algorithm was taken from a program with the name “Pivot de Gauss v1.8” written by Nicolas Girod in 1998 (filename gauss.89p).
His e-mail is: bgirod@club-internet.fr

I changed only the variable names to make it fit in to my program.

The source code is given below to document the extend of non-native coding.

Source Code

```
ewev()
Prgm
setMode("Exact/Approx","AUTO"
)
ClrIO
Local
ewev_tt,ewev_em,ewev_ew,ewev_
n,ewev_nv,ewev_row,ewev_t,ewe
v_ttr,ewev_ttc,ewev_yy,ewev_b
ak,ewev_cur,ewev_k,ewev_p,ewe
v_q,ewev_ttz,ewev_vm,ewev_tyc
,ewev_tyr,ewev_tyf,ewev_tyl,e
wev_dt,ewev_los,ewev_ls,ewev_
mz,ewev_sls,ewev_sot,ewev_sol
Disp  ""
Dialog
 Title  "current folder:
"&getFold()
 Request  "Enter
matrix",ewev_tt
EndDlog

©ewev_tt→ewev_tt
Try
dim(ewev_tt)[1]→ewev_ttr
dim(ewev_tt)[2]→ewev_ttc
If ewev_ttr≠ewev_ttc Then
 Text  "not a quadratic
Matrix, dims must be n*n"
 Goto  ende
EndIf
 Else
Text  "not a matrix"
 Goto  ende
EndTry
newMat(ewev_ttr,ewev_ttc)→ewe
v_em
For  ewev_row,1,ewev_ttr

1→ewev_em[ewev_row,ewev_row]
EndFor

ClrIO
Disp  "Matrix: m"
Pause  ewev_tt
ClrIO
Disp  "Unit Matrix: um*λ"
Pause  ewev_em*λ
ClrIO
Disp  "m-um*λ"
ewev_tt-ewev_em*λ→ewev_vm
Pause  ewev_vm
ClrIO
Disp  "det(m-um*λ)=Ø"
det(ewev_vm)→ewev_dt
Pause  expand(ewev_dt)=Ø
Disp  "→ factored"
Pause  factor(ewev_dt)=Ø
ClrIO
zeros(det(ewev_vm),λ)→ewev_ew
If dim(ewev_ew)<1 Then
 Disp  "the characteristic
polynom",ewev_dt=Ø,"doesn't
have real zeros!","→ no
eigVl","→ no eigVc"
 Pause
 Goto  ende
EndIf
Disp  "eigenvalues"
Pause  "λ="&string(ewev_ew)
For  ewev_t,1,dim(ewev_ew)
 ewev_tt-
ewev_em*ewev_ew[ewev_t]→ewev_
yy
 newMat(ewev_ttr,1)→ewev_nv
 ClrIO
 Disp  "finding
eigenvector","solving:"
 Pause  "(m-
um*λ)x=Ø|λ="&string(ewev_ew[e
wev_t])

Lbl ap
1→ewev_q
1→ewev_p
augment(ewev_yy,ewev_nv)→ewev
_cur
ClrIO
Pause  ewev_cur
dim(ewev_cur)→ewev_di
ewev_di[1]→ewev_k
For
ewev_ttz,1,□(ewev_no,ewev_no,
Ø,ewev_ttr-1)
ewev_cur→ewev_bak
ewev_q+1→ewev_q
If  ewev_q=ewev_k+1 and
ewev_p=ewev_k
Goto  bye
If ewev_cur[ewev_q,ewev_p]>Ø
Then
Disp  "L"&string(ewev_q)&" ?
"&string(ewev_cur[ewev_p,ewev
_p])&" L"&string(ewev_q)&" -
"&string(ewev_cur[ewev_q,ewev
_p])&" L"&string(ewev_p)
Pause
EndIf
If ewev_cur[ewev_q,ewev_p]<Ø
Then
Disp  "L"&string(ewev_q)&" ?
"&string(ewev_cur[ewev_p,ewev
_p])&" L"&string(ewev_q)&" +
"&string(-ewev_cur[ewev_q,ewe
v_p])&" L"&string(ewev_p)
Pause
EndIf
mRow(ewev_bak[ewev_p,ewev_p],
ewev_cur,ewev_q)→ewev_cur
mRowAdd(-ewev_bak[ewev_q,ewev
_p],ewev_cur,ewev_p,ewev_q)→e
wev_cur
If ewev_q=ewev_k Then
ewev_p+1→ewev_q
ewev_p+1→ewev_p
EndIf
ClrIO
Pause  ewev_cur
Lbl  bye
EndFor

Ø→ewev_tyl
Ø→ewev_tyf
For  ewev_tyr,ewev_ttr,1,-1
 For  ewev_tyc,1,ewev_ttr
  If
round(ewev_cur[ewev_tyr,ewev_
tyc],11)≠Ø Then
   1→ewev_tyf
  EndIf
 EndFor
  If ewev_tyf=1 Then
   ewev_tyl+1→ewev_tyl
  EndIf
 Ø→ewev_tyf
EndFor

ClrIO
Disp  "(m-
um*λ)x=Ø|λ="&string(ewev_ew[e
wev_t])
Disp  "number of vectors x:
"&string(ewev_ttr-ewev_tyl)

©======
For  ewev_los,1,ewev_ttr-
ewev_tyl
newMat(ewev_ttr,1)→ewev_sot
For
ewev_mz,ewev_ttr,ewev_tyl+1,-
1
(1-(ewev_ttr-
ewev_mz)+ewev_los-
1)*(-1)^(ewev_los+1)→ewev_sot
[ewev_mz,1]
EndFor
For  ewev_mz,ewev_tyl,1,-1
 Ø→ewev_sls
 For
ewev_ls,ewev_mz+1,ewev_ttr

ewev_sls+ewev_sot[ewev_ls,1]*
ewev_cur[ewev_mz,ewev_ls]→ewe
v_sls
 EndFor

ewev_sls/(-ewev_cur[ewev_mz,e
wev_mz])→ewev_sot[ewev_mz,1]
EndFor

If ewev_los=1 Then
 ewev_sot→ewev_sol
EndIf
If ewev_los>1 and
ewev_los≤ewev_ttr-ewev_tyl
Then

augment(ewev_sol,ewev_sot)→ew
ev_sol
EndIf
If ewev_los=ewev_ttr-ewev_tyl
Then
 Pause  ewev_sol
EndIf

EndFor
©======

EndFor

Lbl  ende
DelVar  ewev_di
ClrIO
Disp  "Eigenvalue/vector
finder","version
1.Ø","released 1Øth of Feb.
2ØØØ","written by Marcel
Brätz","marc@mbraetz.de","htt
p://mbraetz.de"
Pause
DispHome
EndPrgm
```

The red marked source code was taken from the gauss program of Nikolas Girod.