Matrices, Geometry&*Mathematica*

Authors: Bruce Carpenter, Bill Davis and Jerry Uhl ©2001
Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.10 The Spectral Theorem for Symmetric Matrices and the Holy Grail of Matrix Theory

TUTORIALS

T.1) If A is of full rank, then PseudoInverse[A] = $(A^t.A)^{-1}\ A^t$

If A is not of full rank, then the formula fails.

□T.1.a.i) If A is of full rank, then PseudoInverse[A] = $(A^t.A)^{-1}\ A^t$

Here is a random full rank matrix A:

```
hitdim = Random[Integer, {3, 7}];
hangdim = hitdim + Random[Integer, {2, 6}];
A = Table[Random[Real, {-3, 3}], {i, 1, hangdim}, {j, 1, hitdim}];
MatrixForm[A]
```

1.5839	-0.146625	0.77245
-2.92929	-1.67887	0.48568
0.775152	-2.91572	1.21009
1.9621	-2.05011	1.33307
1.35237	-1.00302	-0.000712392
-0.51901	-1.07311	-2.9476
1.42094	2.62745	-2.28168

Check:

```
rank = Length[SingularValues[A][[2]]];
rank == hitdim
```

True

If the last cell returns false, rerun everything.

Now look at Mathematica's calculation of PseudoInverse[A]:

```
MatrixForm[PseudoInverse[A]]
```

0.0769035	-0.145882	0.0483356	0.10369	0.0764657	0.00374139
0.00583533	-0.0567625	-0.12356	-0.0809911	-0.061821	-0.143317
0.0392199	0.00869697	-0.00597901	0.0195501	-0.0415013	-0.244314

And Mathematica's calculation of $(A^t.A)^{-1}\ A^t$.

```
MatrixForm[Inverse[Transpose[A].A].Transpose[A]]
```

0.0769035	-0.145882	0.0483356	0.10369	0.0764657	0.00374139
0.00583533	-0.0567625	-0.12356	-0.0809911	-0.061821	-0.143317
0.0392199	0.00869697	-0.00597901	0.0195501	-0.0415013	-0.244314

They are the same. This is superb evidence that

PseudoInverse[A] = $(A^t.A)^{-1}\ A^t$.

Explain why this is guaranteed for any and all matrices A of full rank.

□Answer:

Go with any full rank matrix A.

Go with any Y in hangdimD.

Xclosest = PseudoInverse[A].Y

means A.Xclosest is closer to Y than any other hit with A.

And because perpendicular distance is the shortest distance, this means

Y - A.Xclosest is perpendicular to the subspace of hangdimD consisting of all hits with A.

In other words (Y - A.Xclosest).(A.X) = 0 for all X's in hitdimD.

In view of the transpose manipulation, this is the same as

$(A^t$.(Y - A.Xclosest)).X = 0 for all X in hitdimD.

The only vector in hitdimD perpendicular to all the X's in hitdimD is {0,0,...,0}.

So {0,0,. . .,0} = A^t .(Y - A.Xclosest) = A^t.Y - A^t.A.Xclosest)).

This is the same as

A^t.Y = A^t.A.Xclosest.

Because A is of full rank, A^t.A is invertible.

Hit both sides with $(A^t.A)^{-1}$ to get

$(A^t.A)^{-1}.A^t.Y = (A^t.A)^{-1}.A^t$.A.Xclosest = Xclosest. = PseudoInverse[A].Y.

Read across to see that

$(A^t.A)^{-1}.A^t.Y$ = PseudoInverse[A].Y.

Because this happens for any Y in hangdimD, this tells you that

$(A^t.A)^{-1}.A^t$ = PseudoInverse[A].

And you're out of here.

□T.1.a.ii) If A is of not of full rank, then the formula fails

Here is a full rank matrix A:

```
hitdim = Random[Integer, {4, 7}];
hangdim = hitdim - Random[Integer, {2, 3}];
A = Table[Random[Real, {-3, 3}], {i, 1, hangdim}, {j, 1, hitdim}];
MatrixForm[A]
```

-0.830151	2.13458	-0.122494	-0.865579	2.31647	-1.63787	-0.1
-2.18671	-1.16921	0.58698	-0.277483	-0.396797	-0.131303	-0.36
1.38945	1.25084	-2.12828	2.63781	-1.09154	-0.676049	-2.1
-1.78314	-0.718986	-1.39437	1.64948	-0.917719	2.40351	2.47
2.333	-2.27985	-0.403292	1.65792	0.502209	0.13317	2.87

Now look atMathematica's calculation of PseudoInverse[A]:

```
MatrixForm[PseudoInverse[A]]
```

-0.122171	-0.28108	-0.0198581	-0.113755	0.0452963
0.0857027	-0.308816	-0.0108682	0.097082	-0.169051
-0.0866326	0.0171748	-0.112627	-0.106134	-0.0148024
0.0715582	0.165482	0.181301	0.0566522	0.123959
0.221477	0.0368908	-0.0207014	-0.0079928	0.109896
-0.164781	-0.284405	-0.108472	0.137095	-0.163085
0.0826505	-0.103117	-0.104242	0.116817	0.091836

And Mathematica's attempt at a calculation of $(A^t.A)^{-1}\ A^t$.

```
MatrixForm[Inverse[Transpose[A].A].Transpose[A]]
```

Inverse::luc : Result for Inverse of badly conditioned matrix ≪1≫ may contain significant numerical errors.

-0.0572173	-0.449795	0.122734	0.0693762	-0.107494
-0.103441	0.717436	-0.514167	-0.60892	1.32746
-2.36827	1.94381	-1.08368	-2.16118	2.31891
-1.37317	2.34747	0.065104	-1.56699	0.880252
-0.330512	0.454838	-0.129161	-0.037903	1.13665
-0.673674	0.0951551	-0.387159	-0.0808682	0.720911
0.277895	-0.112558	-0.0969121	0.169937	-0.086267

The calculation failed.

The upshot: The formula

PseudoInverse[A] = $(A^t.A)^{-1}\ A^t$.

fails for this matrix.

What gives?

□Answer:

The formula

PseudoInverse[A] = $(A^t.A)^{-1}\ A^t$

fails for a very good reason: The matrix A is not of full rank.

```
rank = Length[SingularValues[A][[2]]];
rank == hitdim
```

False

T.2) The Spectral Theorem says all symmetric matrices are diagonalizable. You just go with the eigenvectors for your hanger and aligner frames and the eigenvalues for your stretches.

□T.2.a.i) When you make your aligner frame the same as your hanger frame, you make a symmetric matrix

You can make a symmetric matrix by:

a) Specifying a perpendicular frame which you use for both your aligner frame and your hanger frame.

b) Going with any stretch factors you like (including positive, zero or negative).

Try it and see what you get in 2D:

```
Clear[perpframe, alignerframe, hangerframe];
s = Random[Real, {-1.5, 1.5}];

{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];

{alignerframe[1], alignerframe[2]} = {perpframe[1], perpframe[2]};

{hangerframe[1], hangerframe[2]} = {perpframe[1], perpframe[2]};

{xstretch, ystretch} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]};

aligner = {alignerframe[1], alignerframe[2]};
stretcher = {{xstretch, 0}, {0, ystretch}};
hanger = Transpose[{hangerframe[1], hangerframe[2]}];

A = hanger.(stretcher.aligner);
MatrixForm[A]
```

$$\begin{pmatrix} -1.59665 & -0.037138 \\ -0.037138 & -1.65029 \end{pmatrix}$$

```
A == Transpose[A]
```

True

Why did that work?
Why will this work in any dimension?

Answer:

Well,

A = hanger.stretcher.aligner.

Because the aligner frame is the same as the hanger frame, you are guaranteed that

$\text{hanger}^t = \text{aligner}$

and

$\text{aligner}^t = \text{hanger}.$

And because stretcher is a diagonal matrix,

$\text{stretcher}^t = \text{stretcher}.$

So

$A^t = \text{aligner}^t.\text{stretcher}^t.\text{hanger}^t$

= hanger.stretcher.aligner

= A.

This tells you that A is symmetric.

T.2.a.ii) The Spectral Theorem tells you that all symmetric matrices can be made this way

How does the Spectral Theorem tell you that all symmetric matrices can be made this way?

Answer:

Go with any symmetric matrix A hitting on hitdimD.

The Spectral theorem gives you an orthonormal basis of hitdimD (perpendicular frame spanning all of hitdimD)

$\{X_1, X_2, X_3, X_4, \ldots, X_{\text{hitdim}}\}$

consisting of unit eigenvectors of A with corresponding eigenvalues

$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots, \lambda_{\text{hitdim}}\}$

so that

$A.X_j = \lambda_j X_j$ for all j's.

When you put

aligner = $\{X_1, X_2, X_3, X_4, \ldots, X_{\text{hitdim}}\}$

stretcher = DiagonalMatrix[$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots, \lambda_{\text{hitdim}}\}$]

hanger = aligner^t,

it's automatic that

(hanger.stretcher.aligner). $X_j = \lambda_j X_j$ for all the j's.

Because

$A.X_j = \lambda_j X_j$ for all the j's

and because $\{X_1, X_2, X_3, X_4, \ldots, X_{\text{hitdim}}\}$ is a basis for hitdimD, it's now automatic that

A = hanger.stretcher.aligner.

T.2.a.iii) The Spectral Theorem tells you that all symmetric matrices are diagonalizable

How does the Spectral Theorem tell you that all symmetric matrices are diagonalizable?

Answer:

Part ii) above said that when you go with any symmetric matrix A hitting on hitdimD, the Spectral theorem gives you an orthonormal basis of hitdimD (perpendicular frame spanning all of hitdimD)

$\{X_1, X_2, X_3, X_4, \ldots, X_{\text{hitdim}}\}$

consisting of unit eigenvectors of A with corresponding eigenvalues

$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots, \lambda_{\text{hitdim}}\}$

so that

$A.X_j = \lambda_j X_j$ for all j's.

When you put

aligner = $\{X_1, X_2, X_3, X_4, \ldots, X_{\text{hitdim}}\}$

stretcher = DiagonalMatrix[$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots, \lambda_{\text{hitdim}}\}$]

hanger = aligner^t,

it's automatic that

A = hanger.stretcher.aligner.

Because hanger and aligner are both based on the same frame, it's automatic that

$\text{aligner} = \text{hanger}^{-1}$.

So

$A = \text{hanger.stretcher.hanger}^{-1}$.

Now put

SpannerMatrix = hanger

diagonalmatrix = stretcher

and read off

$A = \text{SpannerMatrix.diagonal.SpannerMatrix}^{-1}$.

This is enough to be able to proclaim that A is diagonalizable.

T.3) Gradient vectors: gradf[x, y] = ∇f[x, y] = $\{\partial_x f[x, y], \partial_y f[x, y]\}$.

Hessian matrices : $H_f[x, y] = \begin{pmatrix} \partial_{\{x,2\}} f[x, y] & \partial_{x,y} f[x, y] \\ \partial_{x,y} f[x, y] & \partial_{\{y,2\}} f[x, y] \end{pmatrix}$.

Local maximizers, minimizers and saddle points of functions

T.3.a) Background: Gradient vectors and Hessian matrices: Using them for function max-min

Take any function f[x, y].
The gradient of f[x, y] at a point {x, y} is

gradf[x, y] = ∇f[x, y] = $\{\partial_x f[x, y], \partial_y f[x, y]\}$

```
Clear[f, gradf, x, y];
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]}
```

$\{f^{(1,0)}[x, y], f^{(0,1)}[x, y]\}$

The derivative of f[x,y] with respect to x is in the first slot.
The derivative of f[x,y] with respect to y is in the second slot.

See the gradient ∇f[x, y] for
f[x, y] = x^2 Sin[y]:

```
f[x_, y_] = x^2 Sin[y];

gradf[x, y]
```

$\{2\,x\,\text{Sin}[y], x^2\,\text{Cos}[y]\}$

See the gradient ∇f[x, y] for
f[x, y] = E^{2x} Cos[y]:

```
Clear[f];
f[x_, y_] = E^(2 x) Cos[y];

gradf[x, y]
```

$\{2\,e^{2x}\,\text{Cos}[y], -e^{2x}\,\text{Sin}[y]\}$

Get it?

Take any function f[x,y].
The Hessian matrix $H_f[x, y]$ of f[x,y] at a point {x,y} is

$$H_f[x, y] = \begin{pmatrix} \partial_{\{x,2\}} f[x, y] & \partial_{x,y} f[x, y] \\ \partial_{x,y} f[x, y] & \partial_{\{y,2\}} f[x, y] \end{pmatrix}$$

```
Clear[f, x, y, H];
    Hf[x_, y_] = ( ∂{x,2} f[x, y]   ∂x,y f[x, y]
                   ∂x,y f[x, y]     ∂{y,2} f[x, y] )
MatrixForm[Hf[x, y]]
```

$\{\{f^{(2,0)}[x, y], f^{(1,1)}[x, y]\}, \{f^{(1,1)}[x, y], f^{(0,2)}[x, y]\}\}$

$$\begin{pmatrix} f^{(2,0)}[x, y] & f^{(1,1)}[x, y] \\ f^{(1,1)}[x, y] & f^{(0,2)}[x, y] \end{pmatrix}$$

First horizontal row of $H_f[x, y]$:
{second derivative of f[x,y] with respect to x,
derivative of f[x,y] first with respect to x and then with respect to y}.
Second horizontal row of $H_f[x, y]$:
{derivative of f[x,y] first with respect to x and then with respect to y,
second derivative of f[x,y] with respect to y}

The hessian matrix is guaranteed to be symmetric.

```
Hf[x, y] == Transpose[Hf[x, y]]
```

True

See the gradient ∇f[x,y] for and the Hessian $H_f[x, y]$ for
f[x, y] = $x^2 + 3\,x\,y + y^2$:

```
f[x_, y_] = 5 x^2 + 3 x y + 4 y^2;

gradf[x, y]

MatrixForm[Hf[x, y]]
```

$\{10\,x + 3\,y, 3\,x + 8\,y\}$

$$\begin{pmatrix} 10 & 3 \\ 3 & 8 \end{pmatrix}$$

See the gradient ∇f[x,y] for and the Hessian $H_f[x, y]$ for
f[x, y] = x^2 Sin[y]:

```
f[x_, y_] = x^2 Sin[y];

gradf[x, y]

MatrixForm[Hf[x, y]]
```

```
{2 x Sin[y], x^2 Cos[y]}
```

$$\begin{pmatrix} 2\,\mathrm{Sin}[y] & 2\,x\,\mathrm{Cos}[y] \\ 2\,x\,\mathrm{Cos}[y] & -x^2\,\mathrm{Sin}[y] \end{pmatrix}$$

See the gradient ∇f[x,y] for and the Hessian $H_f[x, y]$ for
$f[x, y] = E^{2x}$ Cos[y]:

```
Clear[f];
f[x_, y_] = E^(2 x) Cos[y];

gradf[x, y]

MatrixForm[Hf[x, y]]
```

```
{2 e^(2 x) Cos[y], -e^(2 x) Sin[y]}
```

$$\begin{pmatrix} 4\,e^{2x}\,\mathrm{Cos}[y] & -2\,e^{2x}\,\mathrm{Sin}[y] \\ -2\,e^{2x}\,\mathrm{Sin}[y] & -e^{2x}\,\mathrm{Cos}[y] \end{pmatrix}$$

Get it?

The recipe for using the Hessian for function max-min is simple in principle.
You take your function, find $\{x_0, y_0\}$ so that
$\nabla f[x_0, y_0] = \{0, 0\}$.
You are guaranteed that $\{x_0, y_0\}$ is

→ a local minimizer of f[x,y] if both eigenvalues of the Hessian $H_f[x_0, y_0]$ are positive.

→ a local maximizer of f[x,y] if both eigenvalues of the Hessian $H_f[x_0, y_0]$ are negative

→ a saddle point of f[x,y] if $H_f[x_0, y_0]$ has one positive and one negative eigenvalue.

→ Otherwise, you get no information.

□T.3.a.i) Trying it out

Here's a function:

```
Clear[f, x, y, gradf, H];
f[x_, y_] = 0.4 (x^2 + 4 Sin[y^2])
```

```
0.4 (x^2 + 4 Sin[y^2])
```

Analyze some of the points at which the gradient of f[x,y] is {0,0} determining whether they are local maximizers, local minimizers or saddle points.

□Answer:

Here are the gradient and Hessian of f[x,y]:

```
Clear[f, x, y, gradf, H];
f[x_, y_] = 1/2 (x^2 + 4 Sin[y^2]);
```

```
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]}
Hf[x_, y_] = ( ∂{x,2} f[x, y]    ∂x,y f[x, y]
               ∂x,y f[x, y]      ∂{y,2} f[x, y] );

MatrixForm[Hf[x, y]]
```

```
{x, 4 y Cos[y^2]}
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 2\,(2\,\mathrm{Cos}[y^2] - 4\,y^2\,\mathrm{Sin}[y^2]) \end{pmatrix}$$

Check out where the gradient is {0,0}:

```
Solve[gradf[x, y] == 0, {x, y}]
```

```
Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found.
```

$$\left\{\{x \to 0, y \to 0\}, \left\{x \to 0, y \to -\sqrt{\tfrac{\pi}{2}}\right\}, \left\{x \to 0, y \to -i\sqrt{\tfrac{\pi}{2}}\right\}, \left\{x \to 0, y \to i\sqrt{\tfrac{\pi}{2}}\right\}, \left\{x \to 0, y \to \sqrt{\tfrac{\pi}{2}}\right\}\right\}$$

This gives rise to several critical points.

Three of them are :

$$\{x_1, y_1\} = \{0, 0\},\ \{x_2, y_2\} = \left\{0, \sqrt{\tfrac{\pi}{2}}\right\} \text{ and } \{x_3, y_3\} = \left\{0, -\sqrt{\tfrac{\pi}{2}}\right\}.$$

Check them out :

```
{x1, y1} = {0, 0};
Eigenvalues[Hf[x1, y1]]
```

```
{1, 4}
```

```
{x2, y2} = {0, Sqrt[π/2]};

Eigenvalues[Hf[x2, y2]]
```

```
{1, -4 π}
```

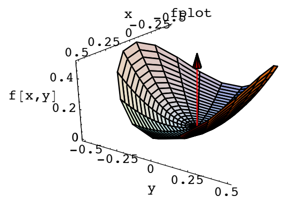
```
{x3, y3} = {0, -Sqrt[π/2]};

Eigenvalues[Hf[x3, y3]]
```

```
{1, -4 π}
```

The Hessian test tells you that $\{x_1, y_1\}$ is a local minimizer and both $\{x_2, y_2\}$ and $\{x_3, y_3\}$ are saddle points.

See a plot of f[x,y] on a circle centered at the local minimizer at $\{x_1, y_1\}$:

```
Clear[x, y, r, s];
{xbase, ybase} = {x1, y1};
x[r_, s_] = xbase + r Cos[s];
y[r_, s_] = ybase + r Sin[s];

fplot =
  ParametricPlot3D[{x[r, s], y[r, s], f[x[r, s], y[r, s]]}, {r, 0, 0.5}
   {s, 0, 2 π}, ViewPoint -> CMView, AxesLabel -> {"x", "y", "f[x,y]"},
   Boxed -> False, PlotLabel -> "fplot", DisplayFunction -> Identity];

Show[fplot, Arrow[{0, 0, 0.5},
   Tail -> {xbase, ybase, f[xbase, ybase]}, VectorColor -> Red],
DisplayFunction -> $DisplayFunction];
```



Yessiree bob! Just as the Hessian predicted. A local minimizer at:

```
{x1, y1}
```

```
{0, 0}
```

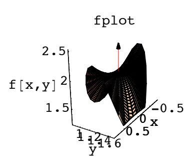
Now check out the saddle point at:

```
{x2, y2}
```

$$\left\{0, \sqrt{\tfrac{\pi}{2}}\right\}$$

Here you go:

```
Clear[x, y, r, s];
{xbase, ybase} = {x2, y2};
x[r_, s_] = xbase + 2 r Cos[s];
y[r_, s_] = ybase + r Sin[s];

fplot =
  ParametricPlot3D[{x[r, s], y[r, s], f[x[r, s], y[r, s]]}, {r, 0, 0.4}
   {s, 0, 2 π}, ViewPoint -> CMView, AxesLabel -> {"x", "y", "f[x,y]"},
   Boxed -> False, PlotPoints -> {30, 30},

   PlotLabel -> "fplot", DisplayFunction -> Identity];

Show[fplot, Arrow[{0, 0, 0.5},
   Tail -> {xbase, ybase, f[xbase, ybase]}, VectorColor -> Red],
DisplayFunction -> $DisplayFunction];
```



Jump on and ride the bronco! Just as the Hessian predicted. A saddle point at:

```
{x2, y2}
```

$$\left\{0, \sqrt{\tfrac{\pi}{2}}\right\}$$

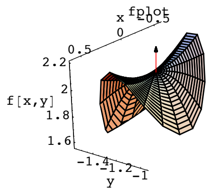
Now check out the saddle point at:

```
{x3, y3}
```

$$\left\{0, -\sqrt{\tfrac{\pi}{2}}\right\}$$

Here you go:

```
Clear[x, y, r, s];
{xbase, ybase} = {x3, y3};

x[r_, s_] = xbase + 2 r Cos[s];
y[r_, s_] = ybase + r Sin[s];

fplot = ParametricPlot3D[{x[r, s], y[r, s], f[x[r, s], y[r, s]]},
    {r, 0, 0.3}, {s, 0, 2 Pi}, ViewPoint -> CMView,
    AxesLabel -> {"x", "y", "f[x,y]"}, Boxed -> False,
    PlotLabel -> "fplot", DisplayFunction -> Identity];

Show[fplot, Arrow[{0, 0, 0.2},
    Tail -> {xbase, ybase, f[xbase, ybase]}, VectorColor -> Red],
DisplayFunction -> $DisplayFunction];
```



Yep! Another saddle point at $\{x_3, y_3\}$.

Just as the Hessian predicted.

□ T.3.a.ii) Why the Hessian test works

Why does the Hessian test work?

□ **Answer:**

Near any point {a, b},

$$\text{approxf}[x, y] = f[a, b] + \nabla f[a, b].\{x - a, y - b\} + \frac{\{x-a, y-b\}.(H_f[a,b].\{x-a, y-b\})}{2}$$

mimics the behavior of f[x, y].

This fact will be explained in the Tutorial immediately below.

Go with a critical point {a,b}. This makes $\nabla f[a,b] = \{0,0\}$, and so the gradient term in approxf[x,y] drops out, leaving

$$\text{approxf}[x, y] = f[a, b] + \frac{\{x-a, y-b\}.(H_f[a,b].\{x-a, y-b\})}{2}$$

Now go with $\{x,y\} \neq \{a,b\}$ and put

$$\{x, y\} = s\,\text{eigvect}[1] + t\,\text{eigvect}[2] + \{a, b\},$$

so that at least one of s and t is not 0.

Here eigvect[1] and eigvect[2] are mutually perpendicular unit eigenvectors of the symmetric matrix $H_f[a, b]$.
with associated eigenvalues eigval[1] and eigval[2] so that

$$H_f[a, b].\text{eigvect}[1] = \text{eigval}[1]\ \text{eigvect}[1]$$

and

$$H_f[a, b].\text{eigvect}[2] = \text{eigval}[2]\ \text{eigvect}[2]$$

This gives

$$\text{approxf}[x, y] = f[a, b] + \frac{(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).H_f[a,b].(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).}{2}$$

Multiply out on the right to get

$$\text{approxf}[x, y] = f[a, b] + \frac{(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).(s\,\text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2}$$

Reason:

$$H_f[a, b].(s\,\text{eigvect}[1]) = s\ \text{eigval}[1]\ \text{eigvect}[1]$$
$$H_f[a, b].(t\,\text{eigvect}[2]) = t\ \text{eigval}[2]\ \text{eigvect}[2]$$

This is the same as

$$\text{approxf}[x, y] = f[a, b] + \frac{s\,\text{eigvect}[1].(s\ \text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2} + \frac{t\,\text{eigvect}[2],.(s\ \text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2}$$

This is the same as

$$\text{approxf}[x, y] = f[a, b] + \frac{s\,\text{eigvect}[1].(s\ \text{eigval}[1]\ \text{eigvect}[1]).}{2} + \frac{t\,\text{eigvect}[2],.(t\ \text{eigval}[2]\ \text{eigvect}[2]),}{2}.$$

Reason:
eigvect[1].eigvect[2] = 0

$$\text{approxf}[x, y] = f[a, b] + \frac{s\,(s\ \text{eigval}[1]).}{2} + \frac{t\,(t\ \text{eigval}[2])}{2}.$$

Reason:
eigvect[1] and eigvect[2] are unit vectors.

This is the same as

$$\text{approxf}[x, y] = f[a, b] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2].$$

Now milk this.

□ Why two positive eigenvalues of $H_f[a, b]$ reveal that {a, b} is a local minimizer of f[x, y]

If eigval[1] and eigval[2] are both positive, then s^2 eigval[1] + t^2 eigval[2] is also positive (because at least one of s and t is not 0) and so

$$\text{approxf}[x, y] = f[a, b] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2]$$
$$= f[a, b] + \text{positive} > f[a, b] = \text{approxf}[a, b]$$

This means {a,b} minimizes approxf[x,y].

And because approxf[x,y] mimics f[x,y] for {x,y} near {a,b}, this tells you that {a,b} is a local minimizer of f[x,y].

□ Why two negative eigenvalues of $H_f[a, b]$ reveal that {a, b} is a local maximizer of f[x, y]

If eigval[1] and eigval[2] are both negative, then s^2 eigval[1] + t^2 eigval[2] is also negative (because at least one of s and t is not 0) and so

$$\text{approxf}[x, y] = f[a, b] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2]$$
$$= f[a, b] + \text{negative} < f[a, b] = \text{approxf}[a, b]$$

This means {a,b} maximizes approxf[a,b].

And because approxf[x,y] mimics f[x,y] for {x,y} near {a,b}, this tells you that {a,b} is a local maximizer of f[x,y].

□ Why one positive eigenvalue and one negative eigenvalue of $H_f[a, b]$ reveal that that the plot of f[x, y] has a saddle point at {a, b, f[a, b]}.

If eigval[1] >0 and eigval[2] < 0 then

$$s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2]$$

is positive for t = 0 and negative for s = 0. (because at least one of s and t is not 0).

When you go with t = 0 and s not zero, you get

$$\text{approxf}[x, y] = f[a, b] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2]$$
$$= f[a, b] + \text{positive} > f[a, b] = \text{approxf}[a, b]$$

When you go with s = 0 and t not zero, you get

$$\text{approxf}[x, y] = f[a, b] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2]$$
$$= f[xx, yy] + \text{negative} < f[a, b] = \text{approxf}[a, b]$$

This means that the plot of approxf[x,y] has a saddle point at {a,b}. And because approxf[x,y] mimics f[x,y] for {x,y} near {a,b}, this tells you that the plot of f[x,y] has a saddle point at {a,b}.

This also tells you that

-> When you leave {a,b} in the direction of eigvect[1], then f[x,y] initially goes up.

-> When you leave {a,b} in the direction of eigvect[2], then f[x,y] initially goes down.

□ T.3.a.iii) Why

$$\textbf{approxf[x,y]} = f[a, b] + \nabla f[a, b].\{x - a, y - b\} + \frac{\{x-a, y-b\}.(H_f[a,b].\{x-a, y-b\})}{2}$$

mimics f[x,y] near {a,b}

Explain this:

Near any point {a, b},

$$\text{approxf}[x,y] = f[a, b] + \nabla f[a, b].\{x - a, y - b\} + \frac{\{x-a, y-b\}.(H_f[a,b].\{x-a, y-b\})}{2}$$

mimics the behavior of f[x, y].

□ **Answer:**

Take any fixed point {a, b} and put

$$\text{approxf}[x, y] =$$
$$f[a, b] + \nabla f[a, b].\{x - a, y - b\} + \frac{\{x - a, y - b\}.(H_f[a, b].\{x - a, y - b\})}{2}$$

Compare f[a, b] and approxf[a, b] :

```
Clear[f, gradf, hessianf, x, y, a, b, xbase, H, approxf, ybase];
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};

Hf[x_, y_] = Hf[x_, y_] = ( ∂{x,2} f[x, y]   ∂x,y f[x, y]
                            ∂x,y f[x, y]    ∂{y,2} f[x, y] );

approxf[x_, y_] =
  f[a, b] + ∇f[a, b].{x - a, y - b} +
   {x - a, y - b}.(Hf[a, b].{x - a, y - b})
   ----------------------------------------  ;
                      2
f[a, b]

approxf[a, b]
```

f[a, b]

∇f[a, b].{0, 0} + f[a, b]

The result :

$$f[a, b] = \text{approxf}[a, b].$$

Now compare the first derivatives at {a, b} :

```
{∂x f[x, y] /. {x -> a, y -> b},
      ∂x approxf[x, y] /. {x -> a, y -> b}}
```

$\{f^{(1,0)}[a, b], 0.\{0, 0\} + \nabla f[a, b].\{1, 0\}\}$

```
{∂_y f[x, y] /. {x -> a, y -> b},
      ∂_y approxf[x, y] /. {x -> a, y -> b}}
{f^(0,1)[a, b], 0.{0, 0} + ∇f[a, b].{0, 1}}
```

The call:

f[x,y] and approxf[x,y] have the same first derivatives at {a,b}.

Compare second derivatives at with respect to x and y at {a,b}:

```
{∂_{x,2} f[x, y] /. {x -> a, y -> b},
    ∂_{x,2} approxf[x, y] /. {x -> a, y -> b}}
{f^(2,0)[a, b], 0.{0, 0} + 2 0.{1, 0} + ∇f[a, b].{0, 0} + f^(2,0)[a, b]}
{∂_{y,2} f[x, y] /. {x -> a, y -> b},
    ∂_{y,2} approxf[x, y] /. {x -> a, y -> b}}
{f^(0,2)[a, b], 0.{0, 0} + 2 0.{0, 1} + ∇f[a, b].{0, 0} + f^(0,2)[a, b]}
{∂_{x,y} f[x, y] /. {x -> a, y -> b},
    ∂_{x,y} approxf[x, y] /. {x -> a, y -> b}}
{f^(1,1)[a, b],
 0.{0, 0} + 0.{0, 1} + 0.{1, 0} + ∇f[a, b].{0, 0} + f^(1,1)[a, b]}
```

The upshot: f[x, y] and approxf[x, y] have order of contact 2 at {a, b}.

This tells you that near any point {a, b},

$$f[a, b] + \nabla f[a, b].\{x - a, y - b\} + \frac{\{x - a, y - b\}.(H_f[a,b].\{x - a, y - b\})}{2}$$

mimics the behavior of f[x, y].

T.4) Quadratic forms

$$f[x, y] = a x^2 + b x y + c y^2 + d x + e y + g$$

Ellipses, hyperbolas and parabolas defined by setting a quadratic form equal to a constant

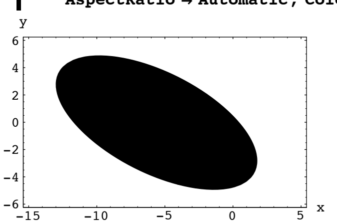
T.4.a.i) A tilted off-set ellipse defined by setting a quadratic form equal to a constant

Here's an example of something folks call a quadratic form:

```
Clear[f, x, y];
f[x_, y_] = 1.1 x^2 + 1.9 x y + 2.5 y^2 + 12.4 x + 10.8 y
12.4 x + 1.1 x^2 + 10.8 y + 1.9 x y + 2.5 y^2
```

And look at this plot:

```
constant = 5;
curveplot = ContourPlot[f[x, y], {x, -15, 5}, {y, -6, 6},
   Contours → {constant}, Axes → True, AxesLabel → {"x", "y"},
   ContourSmoothing → Automatic, PlotPoints → 50,
   AspectRatio → Automatic, ColorFunction → Automatic];
```



The border of the black region is a plot of the curve consisting of all the points {x,y} for which

f[x,y] = constant = 5.

Parameterize and plot this ellipse.

Give the perpendicular frame on which it is hung and measure the length of the long axis and the short axis.

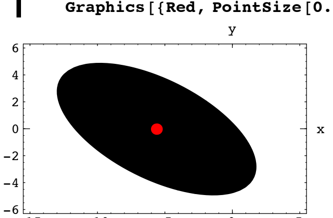
Answer:

Calculate the gradient of f[x,y] and find out where it is {0,0}:

```
Clear[x, y, gradf, H];
gradf[x_, y_] = {∂_x f[x, y], ∂_y f[x, y]};
criticals = Solve[gradf[x, y] == 0];

{xcritical, ycritical} = {x, y} /. criticals[[1]]
{-5.61299, -0.0270636}
```

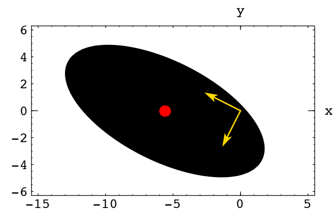
Throw a plot of {xcritical, ycritical} into the mix:

```
embellishedplot = Show[curveplot,
   Graphics[{Red, PointSize[0.04], Point[{xcritical, ycritical}]}]];
```



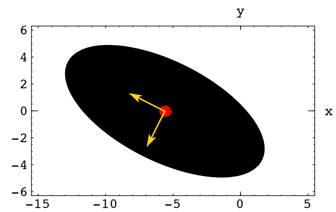
Now calculate a perpendicular frame of eigenvectors of the Hessian of f[x,y] and throw them into the plot:

```
Clear[H];
H_f[x_, y_] = ( ∂_{x,2} f[x, y]   ∂_{x,y} f[x, y]
                ∂_{x,y} f[x, y]   ∂_{y,2} f[x, y] );

Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[H_f[x, y]];

scalefactor = 3;
eigenplot = Table[Arrow[scalefactor eigenvector[k],
    Tail -> {0, 0}, VectorColor -> Gold, HeadSize -> 1], {k, 1, 2}];

setup = Show[embellishedplot , eigenplot];
```



Put the tails of the eigenvectors at {xcritical,ycritical}:

```
adjustedeigenplot = Table[
   Arrow[scalefactor eigenvector[k], Tail -> {xcritical, ycritical},
    VectorColor -> Gold, HeadSize -> 1], {k, 1, 2}];

goodsetup = Show[embellishedplot , adjustedeigenplot];
```



The natural coordinate system for this set up is the {u,v} coordinate system coming from the two plotted unit eigenvectors. This is the perpendicular frame on which the ellipse is hung.

To go from uv-coordinates to xy-coordinates, you just use:

```
Clear[u, v];
{x[u_, v_], y[u_, v_]} =
 {xcritical, ycritical} + u eigenvector[1] + v eigenvector[2]
{-5.61299 - 0.450999 u - 0.892524 v, -0.0270636 - 0.892524 u + 0.450999 v}
```

Look at this:

```
Expand[f[x[u, v], y[u, v]]] == constant
-34.9467 + 2.98004 u^2 + 0.619958 v^2 == 5
```

This is the same as:

```
Expand[f[x[u, v], y[u, v]]] - (f[x[u, v], y[u, v]] /. {u -> 0, v -> 0})
        == constant - (f[x[u, v], y[u, v]] /. {u -> 0, v -> 0})
2.98004 u^2 + 0.619958 v^2 == 39.9467
```

Extract the numerical constants:

```
a = Coefficient[f[x[u, v], y[u, v]], u^2]
2.98004
b = Coefficient[f[x[u, v], y[u, v]], v^2]
0.619958
r = constant - (f[x[u, v], y[u, v]] /. {u → 0, v → 0})
39.9467
```

Now move in with the u-v parametrization:

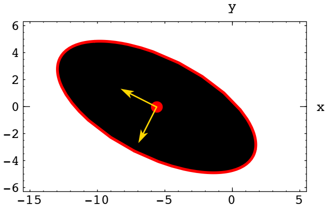
```
Clear[t];
{u[t_], v[t_]} = {√r Cos[t] / √a, √r Sin[t] / √b}
{3.66125 Cos[t], 8.02711 Sin[t]}
```

Now go to the x-y parameterization:

```
Clear[xx, yy];
{xx[t_], yy[t_]} = {x[u[t], v[t]], y[u[t], v[t]]}
{-5.61299 - 1.65122 Cos[t] - 7.16439 Sin[t],
 -0.0270636 - 3.26775 Cos[t] + 3.62022 Sin[t]}
```

See it:

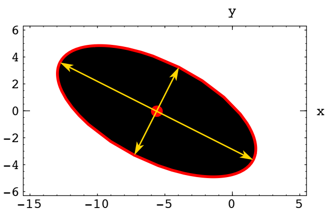
```
ellipseplot = ParametricPlot[{xx[t], yy[t]} ,
   {t, 0, 2 Pi}, PlotStyle -> {{Thickness[0.01], Red}},
   DisplayFunction -> Identity];

Show[goodsetup , ellipseplot];
```



Perfecto.

Make it look pretty:

```
ustretch = Sqrt[r]/Sqrt[a];

vstretch = Sqrt[r]/Sqrt[b];

tail = {xcritical, ycritical};

neweigenplot = {Arrow[ ustretch eigenvector[1],
    Tail -> tail, VectorColor -> Gold, HeadSize -> 1],
        Arrow[- ustretch eigenvector[1], Tail -> tail,
    VectorColor -> Gold, HeadSize -> 1],
        Arrow[ vstretch eigenvector[2], Tail -> tail,
    VectorColor -> Gold, HeadSize -> 1],
        Arrow[- vstretch eigenvector[2], Tail -> tail,
    VectorColor -> Gold, HeadSize -> 1]};

Show[embellishedplot , ellipseplot, neweigenplot ];
```



The lengths of the short and long axes are:

```
{2 Sqrt[r]/Sqrt[a] , 2 Sqrt[r]/Sqrt[b] }
```

```
{7.3225, 16.0542}
```

The ellipse is centered at:

```
{xcritical, ycritical}
```

```
{-5.61299, -0.0270636}
```

□T.4.a.ii) A tilted off-set hyperbola defined by setting a quadratic form equal to a constant

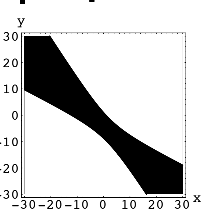
Here's another example of something folks call a quadratic form:

```
Clear[f, x, y];
f[x_, y_] = 1.1 x^2 + 2.9 x y + 1.4 y^2 + 12.4 x + 10.8 y
```

```
12.4 x + 1.1 x^2 + 10.8 y + 2.9 x y + 1.4 y^2
```

And look at this plot:

```
constant = 12;
ranger = 30;

curveplot =
  ContourPlot[f[x, y], {x, -ranger, ranger}, {y, -ranger, ranger},
   Contours → {constant}, Axes → True, AxesLabel → {"x", "y"},
   ContourSmoothing → Automatic, PlotPoints → 50,
   AspectRatio → Automatic, ColorFunction → Automatic];
```



The border of the black region is a plot of the curve consisting of all the points {x,y} for which

f[x,y] = constant = 12.

Parameterize and plot this hyperbola.

Give the perpendicular frame on which it is hung..

□Answer:

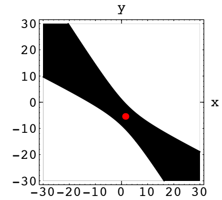
Calculate the gradient of f[x,y] and find out where it is {0,0}:

```
Clear[x, y, gradf, H];
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};
    criticals = Solve[gradf[x, y] == 0];

{xcritical, ycritical} = {x, y} /. criticals[[1]]
```

```
{1.51111, -5.42222}
```

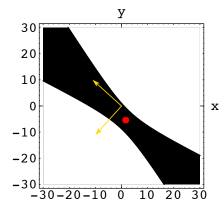
Throw a plot of {xcritical, ycritical} into the mix:

```
embellishedplot = Show[curveplot,
   Graphics[{Red, PointSize[0.04], Point[{xcritical, ycritical}]}]];
```



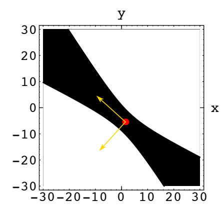
Now calculate a perpendicular frame of eigenvectors of the Hessian of f[x,y] and throw them into the plot:

```
Clear[H];

Hf[x_, y_] = ( ∂{x,2} f[x, y]   ∂x,y f[x, y]
               ∂x,y f[x, y]     ∂{y,2} f[x, y] );
Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[Hf[x, y]];

scalefactor = 15;
eigenplot = Table[Arrow[scalefactor eigenvector[k],
     Tail -> {0, 0}, VectorColor -> Gold, HeadSize -> 3], {k, 1, 2}];

setup = Show[embellishedplot , eigenplot];
```



Put the tails of the eigenvectors at {xcritical,ycritical}:

```
adjustedeigenplot = Table[
   Arrow[scalefactor eigenvector[k], Tail -> {xcritical, ycritical},
    VectorColor -> Gold, HeadSize -> 3], {k, 1, 2}];

goodsetup = Show[embellishedplot , adjustedeigenplot];
```



The natural coordinate system for this set up is the {u,v} coordinate system coming from the two plotted unit eigenvectors. This is the perpendicular frame on which the hyperbola is hung.

To go from uv-coordinates to xy-coordinates, you just use:

```
Clear[u, v];
{x[u_, v_], y[u_, v_]} =
 {xcritical, ycritical} + u eigenvector[1] + v eigenvector[2]
```

```
{1.51111 - 0.669739 u - 0.742597 v, -5.42222 - 0.742597 u + 0.669739 v}
```

Look at this:

```
Expand[f[x[u, v], y[u, v]]] == constant
```

```
-19.9111 + 2.70774 u^2 - 0.207738 v^2 == 12
```

This is the same as:

```
Expand[f[x[u, v], y[u, v]]] - (f[x[u, v], y[u, v]] /. {u -> 0, v -> 0})
     == constant - (f[x[u, v], y[u, v]] /. {u -> 0, v -> 0})
```

```
2.70774 u^2 - 0.207738 v^2 == 31.9111
```

Extract the numerical constants:

```
a = Coefficient[f[x[u, v], y[u, v]], u^2]
```

```
2.70774
```

```
b = Coefficient[f[x[u, v], y[u, v]], v^2]
```

```
-0.207738
```

```
r = constant - (f[x[u, v], y[u, v]] /. {u → 0, v → 0})
```

```
31.9111
```

Now move in with the u-v parameterizations:

```
Clear[u1, v1, u2, v2, t];
{u1[t_], v1[t_]} = {Sqrt[r] Cosh[t]/Sqrt[Abs[a]], Sqrt[r] Sinh[t]/Sqrt[Abs[b]]}
{u2[t_], v2[t_]} = {-Sqrt[r] Cosh[t]/Sqrt[Abs[a]], Sqrt[r] Sinh[t]/Sqrt[Abs[b]]}
```

```
{3.43295 Cosh[t], 12.394 Sinh[t]}
{-3.43295 Cosh[t], 12.394 Sinh[t]}
```

If you want the formulas for Cosh[t] and Sinh[t]
and a little more info about why Sinh[t] and Cosh[t]
are used to parameterize hyperbolas, click on the right.

$$\text{Cosh}[t] = \frac{1}{2}(E^t + E^{-t})$$

$$\text{Sinh}[t] = \frac{1}{2}(E^t - E^{-t})$$

The reason Cos[t] and Sin[t] are used to parametrize ellipses boils down to:

$x = a\sqrt{r}\,\text{Cos}[t]$ and $y = b\sqrt{r}\,\text{Sin}[t]$

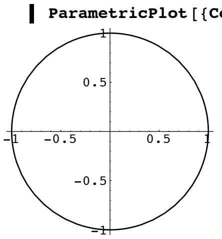
make

$(\frac{x}{a})^2 + (\frac{y}{b})^2 = r$

```
Clear[a, b, r, t];
Simplify[(a Sqrt[r] Cos[t]/a)^2 + (b Sqrt[r] Sin[t]/b)^2]
```

r

```
ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2 Pi}];
```

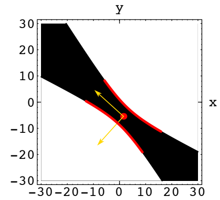


The reason Cosh[t] and Sinh[t] are used to parametrize hyperbolas boils down to:

$x = a\sqrt{r}\,\text{Cosh}[t]$ and $y = b\sqrt{r}\,\text{Sinh}[t]$

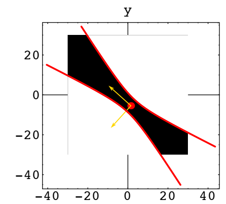
make

$(\frac{x}{a})^2 - (\frac{y}{b})^2 = r$:



See more:

```
{tlow, thigh} = {-2, 2};
branch1plot = ParametricPlot[{x1[t], y1[t]} ,
   {t, tlow, thigh}, PlotStyle -> {{Thickness[0.01], Red}},
   DisplayFunction -> Identity];

branch2plot = ParametricPlot[{x2[t], y2[t]} ,
   {t, tlow, thigh}, PlotStyle -> {{Thickness[0.01], Red}},
   DisplayFunction -> Identity];

niceplot =
  Show[goodsetup , branch1plot , branch2plot, PlotRange -> All ];
```



Lookin' good and feelin' good.

Make it look pretty:

```
a = Coefficient[f[x[u, v], y[u, v]], u^2]
```

```
2.70774
```

```
b = Coefficient[f[x[u, v], y[u, v]], v^2]
```

```
-0.207738
```

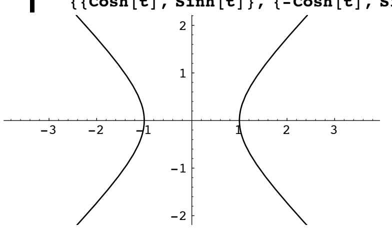
```
r = constant - (f[x[u, v], y[u, v]] /. {u → 0, v → 0})
```

```
31.9111
```

```
Clear[a, b, r, t];
Simplify[(a Sqrt[r] Cosh[t]/a)^2 - (b Sqrt[r] Sinh[t]/b)^2]
```

r

```
ParametricPlot[
  {{Cosh[t], Sinh[t]}, {-Cosh[t], Sinh[t]}}, {t, -2, 2}];
```



That's why

→ Sinh[t] is called the hyperbolic Sine of t.

→ Cosh[t] is called the hyperbolic Cosine of t.

Now go to the x-y parameterizations:

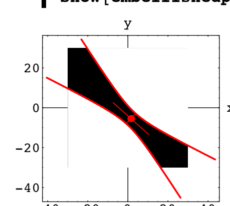
```
Clear[x1, y1, y1, y2];
{x1[t_], y1[t_]} = { x[u1[t], v1[t]], y[u1[t], v1[t]]}
{x2[t_], y2[t_]} = { x[u2[t], v2[t]], y[u2[t], v2[t]]}
```

```
{1.51111 - 2.29918 Cosh[t] - 9.20377 Sinh[t],
 -5.42222 - 2.5493 Cosh[t] + 8.30078 Sinh[t]}
{1.51111 + 2.29918 Cosh[t] - 9.20377 Sinh[t],
 -5.42222 + 2.5493 Cosh[t] + 8.30078 Sinh[t]}
```

See it:

```
{tlow, thigh} = {-1, 1};
branch1plot = ParametricPlot[{x1[t], y1[t]} ,
   {t, tlow, thigh}, PlotStyle -> {{Thickness[0.015], Red}},
   DisplayFunction -> Identity];

branch2plot = ParametricPlot[{x2[t], y2[t]} ,
   {t, tlow, thigh}, PlotStyle -> {{Thickness[0.015], Red}},
   DisplayFunction -> Identity];

Show[goodsetup , branch1plot , branch2plot, PlotRange -> All ];
```

```
ustretch = Sqrt[r]/Sqrt[Abs[a]] ;

vstretch = Sqrt[r]/Sqrt[Abs[b]] ;

tail = {xcritical, ycritical};
 neweigenplot = {Arrow[ ustretch eigenvector[1],
     Tail -> tail, VectorColor -> Red, HeadSize -> 1],
         Arrow[- ustretch eigenvector[1], Tail -> tail,
     VectorColor -> Red, HeadSize -> 1],
         Arrow[ vstretch eigenvector[2], Tail -> tail,
     VectorColor -> Red, HeadSize -> 1],
         Arrow[- vstretch eigenvector[2], Tail -> tail,
     VectorColor -> Red, HeadSize -> 1]};

Show[embellishedplot, branch1plot , branch2plot, neweigenplot ];
```



□ T.4.a.iii) A tilted off-set parabola defined by setting a quadratic form equal to a constant

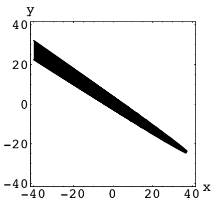
Here's yet another example of something folks call a quadratic form:

```
Clear[f, x, y];
f[x_, y_] = 2.0 x^2 + 6.0 x y + 4.5 y^2 - 2.0 x - 5.0 y
```

$-2.\ x + 2.\ x^2 - 5.\ y + 6.\ x\ y + 4.5\ y^2$

Look at this plot:

```
constant = 50;
ranger = 40;

curveplot =
  ContourPlot[f[x, y], {x, -ranger, ranger}, {y, -ranger, ranger},
   Contours → {constant}, Axes → True, AxesLabel → {"x", "y"},
   ContourSmoothing → Automatic, PlotPoints → 50,
   AspectRatio → Automatic, ColorFunction → Automatic];
```



The border of the black region is a plot of the curve consisting of all the points {x,y} for which

f[x,y] = constant = 50.

Parameterize, identify and plot this curve.
Identify the point at the tip on the right.

Answer:

Copy, paste and edit:

Calculate the gradient of f[x,y] and find out where it is {0,0}:

```
Clear[x, y, gradf, H];
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};
criticals = Solve[gradf[x, y] == 0];

{xcritical, ycritical} = {x, y} /. criticals[[1]]
```

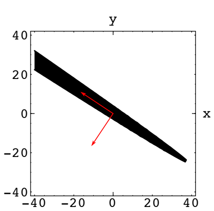
```
RowReduce::luc : Result for RowReduce of badly conditioned matrix
  {{4., 6., -2.}, {6., 9., -5.}} may contain significant numerical errors.
```

$\{6.0048\times10^{15},\ -4.0032\times10^{15}\}$

This time, there are no critical points.

In spite of this little set back, calculate a perpendicular frame of eigenvectors of the Hessian of f[x,y] and throw them into the plot:

```
Clear[H];
Hf[x_, y_] =
  {{∂{x,2} f[x, y], ∂x,y f[x, y]}, {∂x,y f[x, y], ∂{y,2} f[x, y]}};

Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[Hf[x, y]];

scalefactor = 20;
eigenplot = Table[Arrow[scalefactor eigenvector[k],
    Tail -> {0, 0}, VectorColor -> Red, HeadSize -> 3], {k, 1, 2}];

setup = Show[curveplot , eigenplot];
```



This looks promising.

The natural coordinate system for this set up is the {u,v} coordinate system coming from the two plotted unit eigenvectors. This is the perpendicular frame on which the curve is hung.

To go from uv-coordinates to xy-coordinates, you just use:

```
Clear[u, v];
{x[u_, v_], y[u_, v_]} = u eigenvector[1] + v eigenvector[2]
```

$\{-0.5547\,u - 0.83205\,v,\ -0.83205\,u + 0.5547\,v\}$

Look at this:

```
uvequation = Expand[f[x[u, v], y[u, v]]] == constant
```

$5.26965\,u + 6.5\,u^2 - 1.1094\,v == 50$

This exhibits v as a quadratic function of u and reveals that the curve is a parabola!

Solve for v in terms of u:

Mathematica's **Solve** instruction produces something crazy here, so go forward by hand.

```
v[u_] = (5.26965 u + 6.5 u^2 - 50) / 1.1094
```

$0.901388\,(-50 + 5.26965\,u + 6.5\,u^2)$

This is the u-v parametrization with parameter u.:

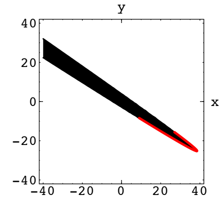
Now go to the x-y parameterization:

```
Clear[xx, yy];
{xx[u_], yy[u_]} = Expand[{ x[u, v[u]], y[u, v[u]]}]
```

$\{37.5 - 4.50694\,u - 4.875\,u^2,\ -25. + 1.80278\,u + 3.25\,u^2\}$

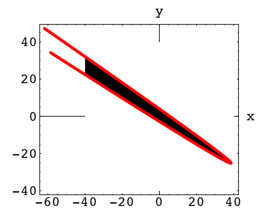
See it:

```
{ulow, uhigh} = {-2, 2};
parabolaplot = ParametricPlot[{xx[u], yy[u]} ,
   {u, ulow, uhigh}, PlotStyle -> {{Thickness[0.015], Red}},
   DisplayFunction -> Identity];

Show[curveplot, parabolaplot];
```



See more:

```
{ulow, uhigh} = {-5, 4};
parabolaplot = ParametricPlot[{xx[u], yy[u]} ,
   {u, ulow, uhigh}, PlotStyle -> {{Thickness[0.015], Red}},
   DisplayFunction -> Identity];

niceplot = Show[curveplot, parabolaplot];
```



Okay.

To try to identify the point at the tip on the right, look at:

```
v[u]
```

$0.901388\,(-50 + 5.26965\,u + 6.5\,u^2)$

Set v'[u] = 0 and solve for u:

```
usol = Solve[v'[u] == 0, u]
```

$\{\{u \to -0.405358\}\}$

In the u-v coordinate system, the tip is at

```
{utip, vtip} = {u, v[u]} /. usol[[1]]
```

$\{-0.405358,\ -46.0321\}$

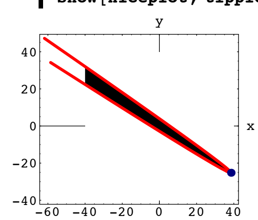
In the x-y coordinate system, the tip is at

```
{xtip, ytip} = {x[utip, vtip], y[utip, vtip]}
```

$\{38.5259,\ -25.1968\}$

Check:

```
tipplot =
  Graphics[{NavyBlue, PointSize[0.04], Point[{xtip, ytip}]}];
Show[niceplot, tipplot];
```



Nailed it.

And you're out of here.

T.4.b) Eigenvalues and eigenvectors of the Hessian explain why the plots in part a) turned out the way they did

The facts behind the work above are:
Go with the quadratic form

$$f[x, y] = a\,x^2 + b\,x\,y + c\,y^2 + d\,x + e\,y + g.$$

Find $\{x_0, y_0\}$ so that

$\text{gradf}[x_0, y_0] = \{0, 0\}$.

Calculate two perpendicular unit eigenvectors, eigvec[1] and eigvec[2], and corresponding eigenvalues,eigval[1] and eigval[2] of the Hessian $H_f[x, y]$

When you set

f[x,y] = constant

and plot the resulting curve, here's what you can get:

Ellipse:

If eigval[1] and eigval[2] are both positive or are both negative, then you get an ellipse centered at $\{x_0, y_0\}$ framed by eigvec[1] and eigvec[2] .

Hyperbola:

If eigval[1] > 0 and eigval[2] < 0 or eigval[1] < 0 and eigval[2] > 0, then you get a hyperbola centered $\{x_0, y_0\}$ and framed by eigvec[1] and eigvec[2] .

Line:

If eigval[1] $\neq$ 0 and eigval[2] = 0 , then you get a line running in the direction of eigvec[2].

Parabola:

If gradf[x , y] = {0, 0} has no solution then you get a parabola framed by eigvec[1] and eigvec[2] .

Explain some of these good facts.

Answer:

Go with

$$f[x, y] = a x^2 + b x y + c y^2 + d x + e y + g.$$

If gradf[x, y] = {0, 0} has a solution $\{x_0 , y_0\}$, then it turns out that

$$f[x, y] = f[x_0 , y_0] + \nabla f[x_0 , y_0].\{x - x_0, y - y_0\} + \frac{\{x - x_0, y - y_0\}.(H_f[x_0 , y_0].\{x - x_0, y - y_0\})}{2}.$$

Check this with *Mathematica*.

```
Clear[f, x, y, a, b, c, d, e, g, gradf, H];
f[x_, y_] = a x^2 + b x y + c y^2 + d x + e y + g;
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};
Hf[x_, y_] = ( ∂{x,2} f[x, y]   ∂x,y f[x, y]
               ∂x,y f[x, y]    ∂{y,2} f[x, y] );
Simplify[ f[x0 , y0] + gradf[x0 , y0].{x - x0, y - y0} +
   ({x - x0, y - y0}.(Hf[x0 , y0].{x - x0, y - y0}))/2 ]
```

$g + d x + a x^2 + e y + b x y + c y^2$

```
f[x, y]
```

$g + d x + a x^2 + e y + b x y + c y^2$

So

$$f[x, y] = f[x_0 , y_0] + \nabla f[x_0 , y_0].\{x - x_0, y - y_0\} + \frac{\{x - x_0, y - y_0\}.(H_f[x_0 , y_0].\{x - x_0, y - y_0\})}{2}.$$

And because $\nabla f[x_0 , y_0] = \{0,0\}$, this simplifies to

So

$$f[x, y] = f[x_0 , y_0] + \frac{\{x - x_0, y - y_0\}.(H_f[x_0 , y_0].\{x - x_0, y - y_0\})}{2}.$$

Go with $\{x,y\} \neq \{x_0 , y_0\}$ and put

$\{x, y\} = s\,\text{eigvect}[1] + t\,\text{eigvect}[2] + \{x_0 , y_0\}$,

so that at least one of s and t is not 0.

Here eigvect[1] and eigvect[2] are mutually perpendicular unit eigenvectors of the symmetric matrix $H_f[x_0 , y_0]$.
with associated eigenvalues eigval[1] and eigval[2] so that

$H_f[x_0 , y_0].\text{eigvect}[1] = \text{eigval}[1]\ \text{eigvect}[1]$

and

$H_f[x_0 , y_0].\text{eigvect}[2] = \text{eigval}[2]\ \text{eigvect}[2]$

This gives

$$f[x, y] = f[x_0 , y_0] + \frac{(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).H_f[x_0 ,y_0].(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).}{2}$$

Multiply out on the right to get

$$f[x, y] = f[x_0 , y_0] + \frac{(s\,\text{eigvect}[1] + t\,\text{eigvect}[2],).(s\ \text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2}$$

Reason:

$H_f[f[x_0 , y_0]].(s\,\text{eigvect}[1]) = s\ \text{eigval}[1]\ \text{eigvect}[1]$

$H_f[f[x_0 , y_0]].(t\,\text{eigvect}[2]) = t\ \text{eigval}[2]\ \text{eigvect}[2]$

This is the same as

$$f[x, y] = f[x_0 , y_0] + \frac{s\,\text{eigvect}[1].(s\ \text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2} + \frac{t\,\text{eigvect}[2],.(s\ \text{eigval}[1]\ \text{eigvect}[1] + t\ \text{eigval}[2]\ \text{eigvect}[2],).}{2}$$

This is the same as

$$f[x, y] = f[x_0 , y_0] + \frac{s\,\text{eigvect}[1].(s\ \text{eigval}[1]\ \text{eigvect}[1]).}{2} + \frac{t\,\text{eigvect}[2],.(t\ \text{eigval}[2]\ \text{eigvect}[2]),}{2}.$$

Reason:
eigvect[1].eigvect[2] = 0

And

$$f[x, y] = f[x_0 , y_0] + \frac{s\ (s\ \text{eigval}[1]).}{2} + \frac{t\ (t\ \text{eigval}[2])}{2}.$$

Reason:
eigvect[1] and eigvect[2] are unit vectors.

This is the same as

$$f[x, y] = f[x_0 , y_0] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2].$$

Now milk this.

When you set f[x,y] = constant, you get:

$$f[x_0 , y_0] + s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2] = \text{constant}$$

This is the same as:

$$s^2\ \text{eigval}[1] + t^2\ \text{eigval}[2] = \text{constant} - f[x_0 , y_0]$$

In the coordinate system coming from eigvec[1] and eigvect[2] with tails at $\{x_0 , y_0\}$, this gives:

▪Ellipses centered at {s,t} = {0,0} if eigval[1] and eigval[2] are both positive or both negative.

When you go back to x-y coordinates, these ellipse are centered at the critical point $\{x_0 , y_0\}$ and framed by eigvec[1] and eigvec[2].

▪ Hyperbolas centered at {s,t} = {0,0} if eigval[1] > 0 and eigval[2] < 0 or if eigval[1] < 0 and eigval[2] > 0

When you go back to x-y coordinates, these hyperbolas are centered at the critical point $\{x_0 , y_0\}$and framed by eigvec[1] and eigvec[2].

▪ Lines defined by

$$s = +\sqrt{\frac{\text{constant} - f[x_0,x_0]}{\text{eigval}[1]}} \quad \text{and } s = -\sqrt{\frac{\text{constant} - f[x_0,x_0]}{\text{eigval}[1]}}$$

if eigval[1] $\neq$ 0 and eigval[2] = 0.When you go back to x-y coordinates, these lines run parallel to eigvec[2].

If there is no solution of $\nabla f[x_0 , y_0] = \{0, 0\}$, then f[x,y] has no maximum, no minimum and no saddle point.

The result: The plot of f[x,y] is an infinite mountain (or valley) whose cross sections are parabolas. (A more detailed explanation of the parabola case is possible but requires lots of algebra.)