$$(\underline{x}-\underline{\mu}_1)^T \Sigma_1 (\underline{x}-\underline{\mu}_1) + (\underline{x}-\underline{\mu}_2)^T \Sigma_2 (\underline{x}-\underline{\mu}_2)$$

~~summation~~

Σ_1 and Σ_2 are just another matrices

Visualizing eigen vectors

Helper functions

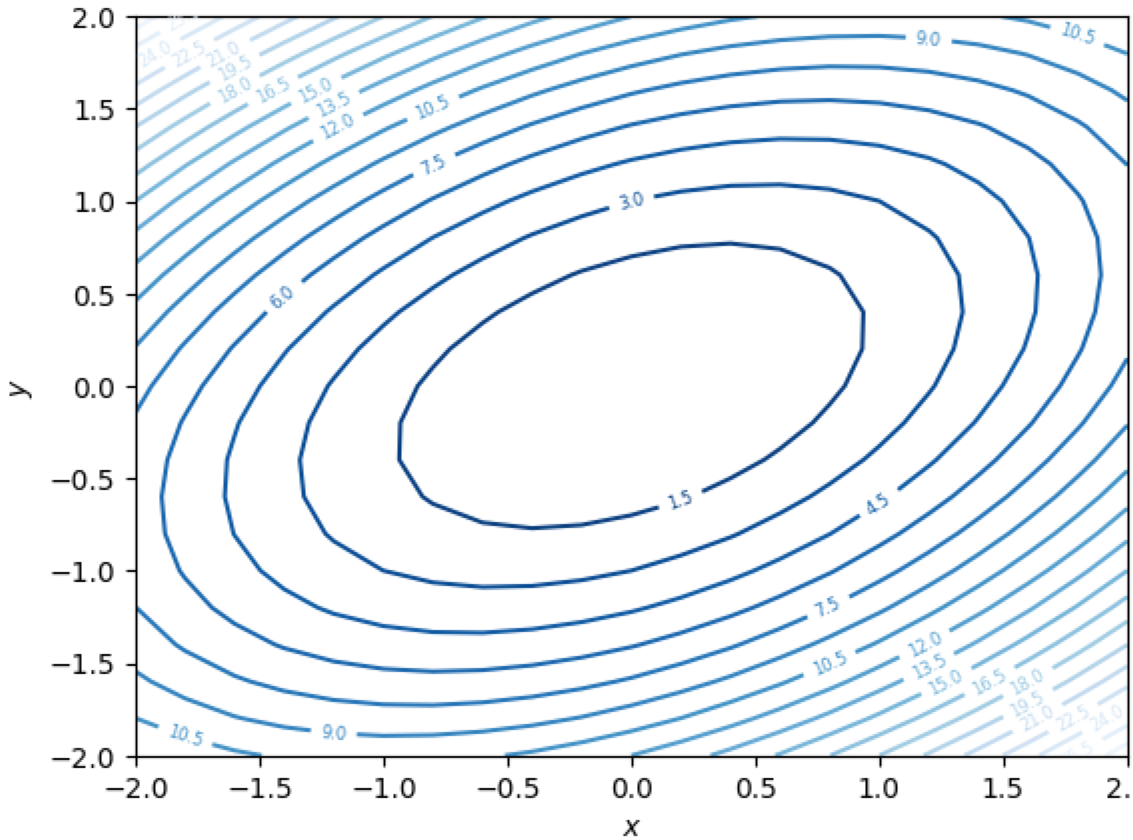
Latex macros

In [1]:
```python
import numpy as np
def random_symmetric_matrix(n):
    A = np.random.rand(n, n)
    return A + A.T

def random_positive_definite_matrix(n):
    A = np.random.rand(n, n)
    return A.T @ A
```

In [2]:
```python
import matplotlib.pyplot as plt
def plot_contour(ax, func):
    x, y = np.mgrid[-2:2:21j,
                    -2:2:21j]
    bfx = np.concatenate([x[..., None], y[..., None]], axis=-1)
    f = func(bfx)
    ctr = ax.contour(x, y, f, 20, cmap='Blues_r')
    plt.clabel(ctr, ctr.levels, inline=True, fontsize=6)
    ax.set_xlabel('$x$')
    ax.set_ylabel('$y$')
    return ax


def draw_arrows(ax, xs):
    ax.arrow(0, 0, xs[0, 0], xs[0, 1], color='r', head_width=0.05)
    ax.arrow(0, 0, xs[1, 0], xs[1, 1], color='y', head_width=0.05)
    ax.axis('equal')
    return ax
```

In [3]:
```python
def f(x, A=np.array([[2, -1], [-1, 3]])):
    # x is m x m x n tensor
    x_row_vec = x[..., None, :] # m x m x 1 x n
    x_col_vec = x[..., :, None] # m x m x n x 1
    res = x_row_vec @ A @ x_col_vec # m x m x 1 x 1
    return res[..., 0, 0] # m x m

fig, ax = plt.subplots()
plot_contour(ax, f)
ax
```

Out[3]:
```
<Axes: xlabel='$x$', ylabel='$y$'>
```



Eigeven value decomposition

Recall that eigen values $\lambda \in \mathbb{R}$ and eigen vectors $\mathbf{v} \in \mathbb{R}^n$ are the solutions to the equation,

$$\underline{A\mathbf{v} = \lambda\mathbf{v}}$$

There are n such solutions let λ_i $\mathbf{v}_i$ for $i \in \{1, \dots, n\}$ be such solutions.

$$\begin{aligned} A\mathbf{v}_1 &= \lambda_1\mathbf{v}_1 && (1) \\ A\mathbf{v}_2 &= \lambda_2\mathbf{v}_2 && (2) \\ &\vdots && (3) \\ A\mathbf{v}_n &= \lambda_n\mathbf{v}_n && (4) \end{aligned}$$

You can arrange these equations in a matrix

$$\begin{bmatrix} A\mathbf{v}_1 & A\mathbf{v}_2 & \dots & A\mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \dots & \lambda_n\mathbf{v}_n \end{bmatrix}$$

You can take A common from the left hand side. On the right hand side, you will have to construct a diagonal matrix of λ_i for factorizing eigen vectors and eigen values.

$$A\underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}}_{\Lambda}$$

HW4: Problem 2

eigen basis

Verify that the above matrix multiplication gives the same results.

$$\boxed{AV = V\Lambda}$$

This is true for all square matrix

V is a matrix of eigen vectors arranged as columns. And Λ is a diagonal matrix of eigen values.

A is symmetric

If the matrix V is invertible, then we can right multiply V^{-1} to both sides,

$$\boxed{A = V\Lambda V^{-1}}$$

Eigen value decomposition

We have found a way to factorize (or decompose) a square matrix A. This particular decomposition is called *Eigen value decomposition*. When such a decomposition exists, the matrix A is called diagonalizable. A generalized form of such decomposition also exists, which is called generalized eigen value decomposition, where the middle matrix is a block diagonal matrix, not diagonal matrix.

Theorem: Symmeteric matrices have orthonormal Eigen vectors

When a square matrix $A \in \mathbb{R}^{n\times n}$ is real and symmetric, then its Eigen value decomposition exists, A is diagonalizable, and the eigen matrix V is an orthogonal matrix.

If the matrix A is symmetric,

$$A = V\Lambda V^\top$$

where $V^{-1} = V^\top$.

Proof:

Proof will take too much time out of the class. Interested readers are encouraged to read about Gram-Schmidt orthogonalization, Schur's lemma and Spectral theorem.

Definition (Orthonormal)

A matrix $V \in \mathbb{R}^{n\times n}$ is called orthonormal or orthogonal if $V^\top V = I_n$ equivalently, $V^{-1} = V^\top$.

$$A\underline{v} = \lambda \underline{v}$$

$\downarrow$ always

$$AV = V\Lambda$$

$$V = \begin{bmatrix} \underline{v}_1 & \cdots & \underline{v}_n \end{bmatrix}_{n\times n}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}_{n\times n}$$

V is not invertible

$\times$

Jordan form

etc

Generalized EVD

if A is symmetric

V is invertible

$\downarrow$

$$A = V\Lambda V^{-1}$$ A is diagonalizable

$\boxed{\text{A is symmetric} \subset \text{A is digonalyzable}}$

A is symmetric

$$A^T = A$$

$\downarrow$

$$V^{-1} = V^T$$

V is orthogonal

$\downarrow$

$$\boxed{\text{if } A^T = A \xrightarrow[\text{theorem}]{\text{spectral}} A = V\Lambda V^T}$$

$$f(\underline{x}) = \underline{x}^T A \underline{x}$$

$$= \underline{x} V \Lambda V^T \underline{x}$$

$$= (V^T \underline{x})^T \Lambda \underbrace{(V^T \underline{x})}_{\underline{y}}$$

$$\underline{y} = V^T \underline{x}$$

$$= \underline{y}^T \Lambda \underline{y}$$

$$= \begin{bmatrix} y_1, & y_2 & \cdots & y_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$= \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2$$

$$\lambda_i = \operatorname{sign}(\lambda_i) \left(\sqrt{\lambda_i}\right)^2$$

$$= \underline{y}^T \underbrace{\operatorname{sign}(\Lambda)}_{\text{Diagonal}} \left(\sqrt{\Lambda}\right)^2 \underline{y}$$

square root and square are element-wise

$$= \underline{z}^T \operatorname{sign}(\Lambda) \underline{z}$$

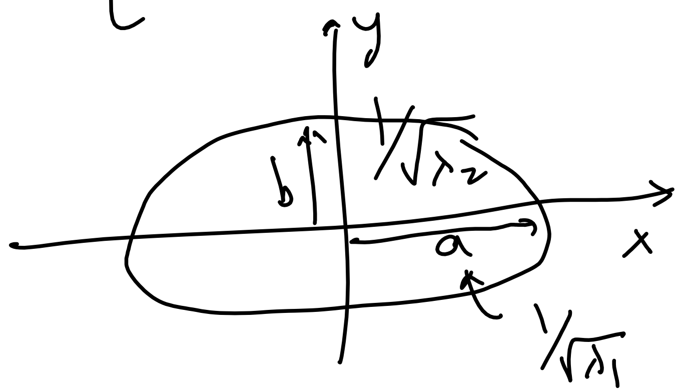
$$\underline{z} = \sqrt{\Lambda}\, \underline{y}$$

$$f_{\underline{z}}(\underline{z}) = \underline{z}^T \,\text{sign}(\underline{\Lambda})\, \underline{z} \qquad \text{sign}(\underline{\Lambda}) = 1$$

$$= \underbrace{z_1^2 + z_2^2 + \dots z_n^2 = f_1}_{\text{eqn of a sphere}}$$

$$f_y(\underline{y}) = \underbrace{\lambda_1 y_1^2 + \dots + \lambda_2 y_n^2 = f_1}_{\text{eqn of an ellipsoid}}$$

In 2D eqn of ellipse is $\left[\dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1\right.$

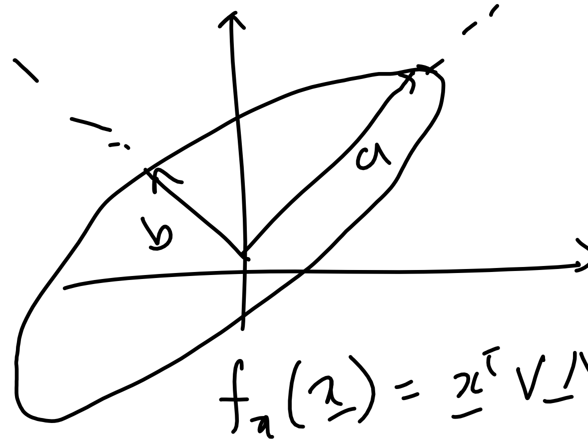


$$f_{\underline{x}}(\underline{x}) = (V^T\underline{x})^T \underline{\Lambda} (\underbrace{V^T\underline{x}}_{\underline{y}})$$

$\underbrace{\text{eigen basis } V \text{ is orthogonal}}$ $(V^{-1} = V^T)$

Rotation $\det(V) = +1$

Reflection $\det(V) = -1$



$f_A(\underline{x}) = \underline{x}^T V \underline{\Lambda} V \underline{x}$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$ Ellipse in 2D

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$ Hyperbola in 2D

PD $\iff$ all eigen values are +ve

$A > 0 \iff \lambda_1, \lambda_2 \ldots \lambda_n > 0$

$\Updownarrow$

$\underline{x}^T A \underline{x}$

$$= x^T V \underline{\Lambda} V^{-1} \underline{x}$$

$$= x^T [\underline{v}_1\ \underline{v}_2 \ldots \underline{v}_n] \underline{\Lambda} [\underline{v}_1\ \underline{v}_2 \quad \underline{v}_n]^{-1} \underline{x}$$

$$= [\underline{x}^T \underline{v}_1\ \ \underline{x}^T v_2 \ldots \underline{x}^T \underline{v}_n] \underline{\Lambda} [\quad]^{-1} \underline{x}$$

$$V^{-1} = [\underline{u}_1, \ldots \underline{u}_n]$$

$$= \begin{bmatrix} \underline{x}^T \underline{v}_1 & \underline{x}^T \underline{v}_2 & \cdots & \underline{x}^T \underline{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} \underline{u}_1 & \cdots & \underline{u}_n \end{bmatrix} \underline{x}$$

$$= \lambda_1 (\underline{x}^T \underline{v}_1)(\underline{u}_1 \underline{x}) + \lambda_2 (\underline{x}^T \underline{v}_2)(\underline{u}_2 \underline{x}) + \cdots + \lambda_n (\underline{x}^T \underline{v}_n)(\underline{u}_n \underline{x})$$

If $\underline{u}_1 = \underline{v}_1^T$ $\qquad V^{-1} = V^T = [\underline{v}_1^T, \ldots \underline{v}_n^T]$

$\underline{u}_2 = \underline{v}_2^T$

$$= \lambda_1 (\underline{x}^T \underline{v}_1)^2 + \lambda_2 (\underline{x}^T \underline{v}_2)^2 + \cdots + \lambda_n (\underline{x}^T \underline{v}_n)^2$$

+ve

if $\lambda_1, \lambda_2, \lambda_n > 0$

then their summation is +ve as well

$\Rightarrow \underline{x}^T A \underline{x} > 0 \Rightarrow A$ is P.D

Example

Consider the quadratic form of a symmetric matrix $A \in \mathbb{R}^{n\times n}$,

$$f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$$

Because the matrix A **is symmetric**, we can write

$$f(\mathbf{x}) = \mathbf{x}^\top V\Lambda V^\top \mathbf{x} = (V\mathbf{x})^\top \Lambda (V\mathbf{x})$$

for eigen value decomposition of $A = V\Lambda V^\top$

In [4]:
```
A = np.array([[2, -1],
              [-1, 3]])
A
```

Out[4]:
```
array([[ 2, -1],
       [-1,  3]])
```

`np.linalg.eigh` finds Eigen values of a Hermitian matrix.

In [5]:
```
lambdas, V = np.linalg.eigh(A)
```

In [6]:
```
V
```

Out[6]:
```
array([[-0.85065081, -0.52573111],
       [-0.52573111,  0.85065081]])
```

Definition (Conjugate transpose)

Conjugate transpose of a complex square matrix $A \in \mathbb{C}^{n\times n}$ is defined as

$$A^H = Re(A)^\top - \iota Im(A)^\top,$$

where $Re(A)$ is the real part of the matrix A and $Im(A)$ is the imaginary part of matrix $A = Re(A) + \iota Im(A)$ and $\iota = \sqrt{-1}$.

Conjugate transpose is the generalization of transpose to complex matrices.

Definition (Hermitian matrix)

A complex matrix $A \in \mathbb{C}^{n\times n}$ is said to be Hermitian matrix if

$$A^H = A$$

Hermitian matrix is a generalization of symmetric matrices to complex matrices.

Theorem: Eigen values of an Hermitian matrices are real

For a complex vector $\mathbf{x} \in \mathbb{C}^n$, the dot product $\mathbf{x}^H\mathbf{x} \in \mathbb{R}$ is real.

$$\mathbf{x}^H\mathbf{x} = \bar{x}_1x_1 + \bar{x}_2x_2 + \cdots + \bar{x}_nx_n,$$

where $\bar{x}_i$ denotes the complex conjugate of x_i. If $x_i = a_i + \imath b_i$, then $\bar{x}_i = a_i - \imath b_i$. Note that $\bar{x}_i x_i = a_i^2 + b_i^2$ is real. Because each element is real, hence their sum $\mathbf{x}^H\mathbf{x}$ is real.

Let $A \in \mathbb{C}^{n\times n}$ be Hermitian then, its eigen vectors $\mathbf{v}_i \in \mathbb{C}^n$ and eigen values $\lambda \in \mathbb{C}$ satisify

$$A\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

Multiply both sides on the left by $\mathbf{v}_i^H$

$$\mathbf{v}_i^H A\mathbf{v}_i = \lambda_i\mathbf{v}^H\mathbf{v}_i$$

or

$$\lambda_i = \frac{\mathbf{v}_i^H A\mathbf{v}_i}{\mathbf{v}^H\mathbf{v}_i}$$

Take complex conjugate on both sides,

$$\bar{\lambda}_i = \frac{\overline{\mathbf{v}_i^H A\mathbf{v}_i}}{\overline{\mathbf{v}_i^H\mathbf{v}_i}}$$

We know the denominator $\mathbf{v}_i^H\mathbf{v}_i$ is real. Numerator can be computed by taking the Hermitian transpose of the product,

$$\bar{\lambda}_i = \frac{\mathbf{v}_i^H A^H\mathbf{v}_i}{\mathbf{v}_i^H\mathbf{v}_i}$$

Because A^H is Hermitian, $A^H = A$, which implies,

$$\bar{\lambda}_i = \lambda_i.$$

This implies that $\lambda_i \in \mathbb{R}$ is real.

When λ_i is real

Theorem: Eigen vectors of a real symmetric matrices are real

Because real symmetric matrices are also hermitian matrices, their eigen values are real. If their eigen values are real, then eigen vectors that are non-zero solutions to the equation

$$(A - \lambda I_n)\mathbf{v} = \mathbf{0}$$

are also real.

`np.linalg.eigh` finds Eigen values of a Hermitian matrix.

In [7]:
```
lambdas, V = np.linalg.eigh(A)
```

In [8]:
```
lambdas
```

Out[8]:
```
array([1.38196601, 3.61803399])
```

In [9]:
```
V
```

Out[9]:
```
array([[-0.85065081, -0.52573111],
       [-0.52573111,  0.85065081]])
```

You can verify that the eigen values and vectors are correct by checking if

$$A = V\Lambda V^\top$$

In [10]:
```
V @ np.diag(lambdas) @ V.T
```

Out[10]:
```
array([[ 2., -1.],
       [-1.,  3.]])
```

In [11]:
```
np.allclose(A, V @ np.diag(lambdas) @ V.T)
```

Out[11]:
```
True
```

Recall that we were analyzing the function,

$$f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} = \mathbf{x}^\top V\Lambda V^\top\mathbf{x} = (V^\top\mathbf{x})^\top\Lambda(V^\top\mathbf{x})$$

With this we can break down the computation of function $f(\mathbf{x})$ in two steps,

$$\mathbf{y} = V^\top\mathbf{x} \iff \mathbf{x} = V\mathbf{y},$$

followed by scaling,

$$f(\mathbf{x}) = \mathbf{y}^\top\Lambda\mathbf{y}$$

If all eigen values are positive (A is positive definite), we can break down eigen values into their square roots and then

$$\mathbf{z} = \sqrt{\Lambda}\mathbf{y} \iff \mathbf{y} = \sqrt{\Lambda}^{-1}$$

can be understood as element wise scaling.

$$f(\mathbf{x}) = \mathbf{z}^\top\mathbf{z}$$

Note that $\mathbf{z}^\top\mathbf{z} = z_1^2 + z_2^2 + \cdots + z_n^2 = f_1$ is equation of a circle in terms of $\mathbf{z}$ for a contour where $f(\mathbf{z}) = f_1$. We can also start from a circle plot for $\mathbf{z}$.

and then compute and plot

$$\mathbf{y} = \sqrt{\Lambda}^{-1}\mathbf{z}.$$

Note that the principle axes of the circle are scaled by eigen values.

Next we compute and plot $\mathbf{x}$ from $\mathbf{y}$

$$\mathbf{x} = V\mathbf{y}$$

In [12]:

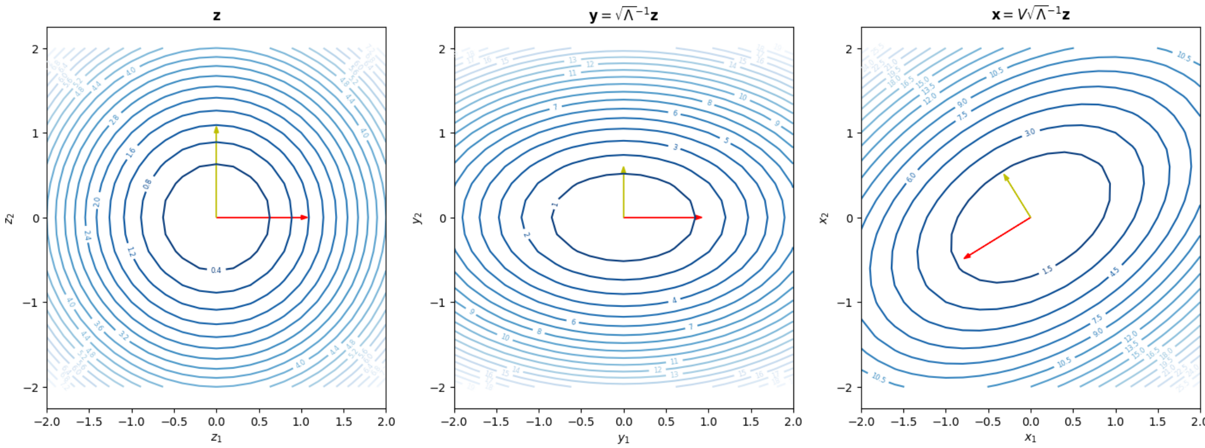
```python
import matplotlib as mpl
from functools import partial

def f_wrt_y(lambdas, y):
    # lambdas is of shape 2
    # y is of shape m x m x n
    return (y * lambdas * y).sum(axis=-1)

def f_wrt_z(z):
    # z is m x m x n
    return (z * z).sum(axis=-1)

zs = np.array([
    [1, 0],
    [0, 1]])
```

In [13]:

```python
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

ax = axes[0]
plot_contour(ax, f_wrt_z)
ax.set_title(r'$\mathbf{z}$')
ax.set_xlabel(r'$z_1$')
ax.set_ylabel(r'$z_2$')
draw_arrows(ax, zs)

ax = axes[1]
plot_contour(ax, partial(f_wrt_y, lambdas))
ax.set_title(r'$\mathbf{y} = \sqrt{\Lambda}^{-1} \mathbf{z}$')
ax.set_xlabel(r'$y_1$')
ax.set_ylabel(r'$y_2$')
ys =  zs / np.sqrt(lambdas)
draw_arrows(ax, ys)

ax = axes[2]
plot_contour(ax, partial(f, A=A))
ax.set_title(r'$\mathbf{x} = V \sqrt{\Lambda}^{-1} \mathbf{z}$')
ax.set_xlabel(r'$x_1$')
ax.set_ylabel(r'$x_2$')
xs = (V @ ys.T).T
draw_arrows(ax, xs)
```

Out[13]:

```
<Axes: title={'center': '$\\mathbf{x} = V \\sqrt{\\Lambda}^{-1} \\mathbf{z}
$'}, xlabel='$x_1$', ylabel='$x_2$'>
```



Note that countours of $f(\mathbf{x})$ are exactly alinged with the transformed circle from $\mathbf{z}$ to $\mathbf{x}$

Eigen values of saddle point (Indefinite matrix)

In [14]:
```
A = np.array([[2, -1],
              [-1, -3]])
A
```

Out[14]:
```
array([[ 2, -1],
       [-1, -3]])
```

In [15]:
```
lambdas, V = np.linalg.eigh(A)
lambdas
```

Out[15]:
```
array([-3.1925824,  2.1925824])
```

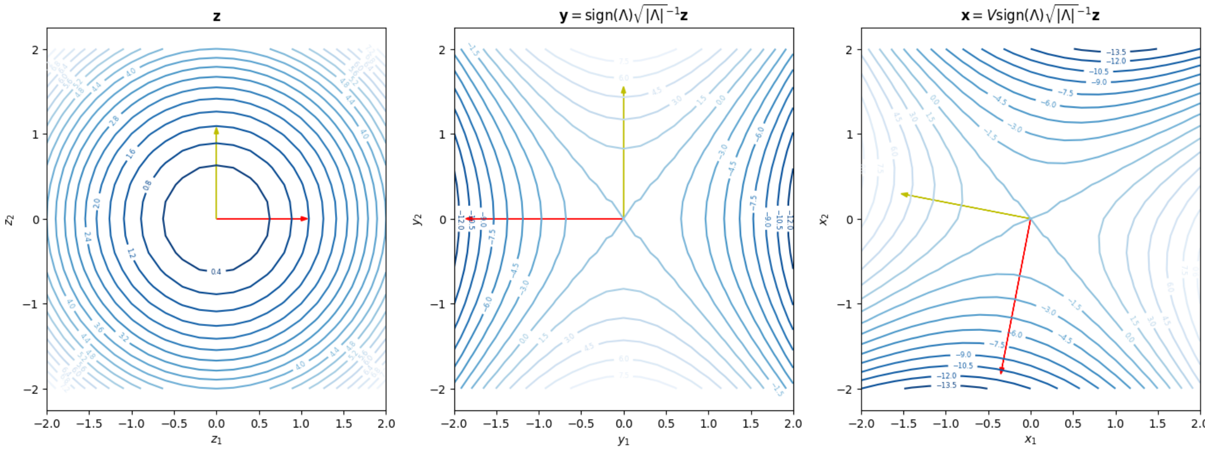
Note that the eigen values are neither all positive nor all negative. We cannot take the square root directly and we have to be respectful of the sign of the lambdas.

In [16]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

ax = axes[0]
plot_contour(ax, f_wrt_z)
ax.set_title(r'$\mathbf{z}$')
ax.set_xlabel(r'$z_1$')
ax.set_ylabel(r'$z_2$')
draw_arrows(ax, zs)

ax = axes[1]
plot_contour(ax, partial(f_wrt_y, lambdas))
ax.set_title(r'$\mathbf{y} = \text{sign}(\Lambda)\sqrt{|\Lambda|}^{-1} \math
ax.set_xlabel(r'$y_1$')
ax.set_ylabel(r'$y_2$')
ys =  zs / np.sign(lambdas) * np.sqrt(np.abs(lambdas))
draw_arrows(ax, ys)

ax = axes[2]
```

```python
plot_contour(ax, partial(f, A=A))
ax.set_title(r'$\mathbf{x} = V \text{sign}(\Lambda)\sqrt{|\Lambda|}^{-1} \ma
ax.set_xlabel(r'$x_1$')
ax.set_ylabel(r'$x_2$')
xs = (V @ ys.T).T
draw_arrows(ax, xs)
```

Out[16]:

```
<Axes: title={'center': '$\\mathbf{x} = V \\text{sign}(\\Lambda)\\sqrt{|\\Lambda|}^{-1} \\mathbf{z}$'}, xlabel='$x_1$', ylabel='$x_2$'>
```



In [ ]: