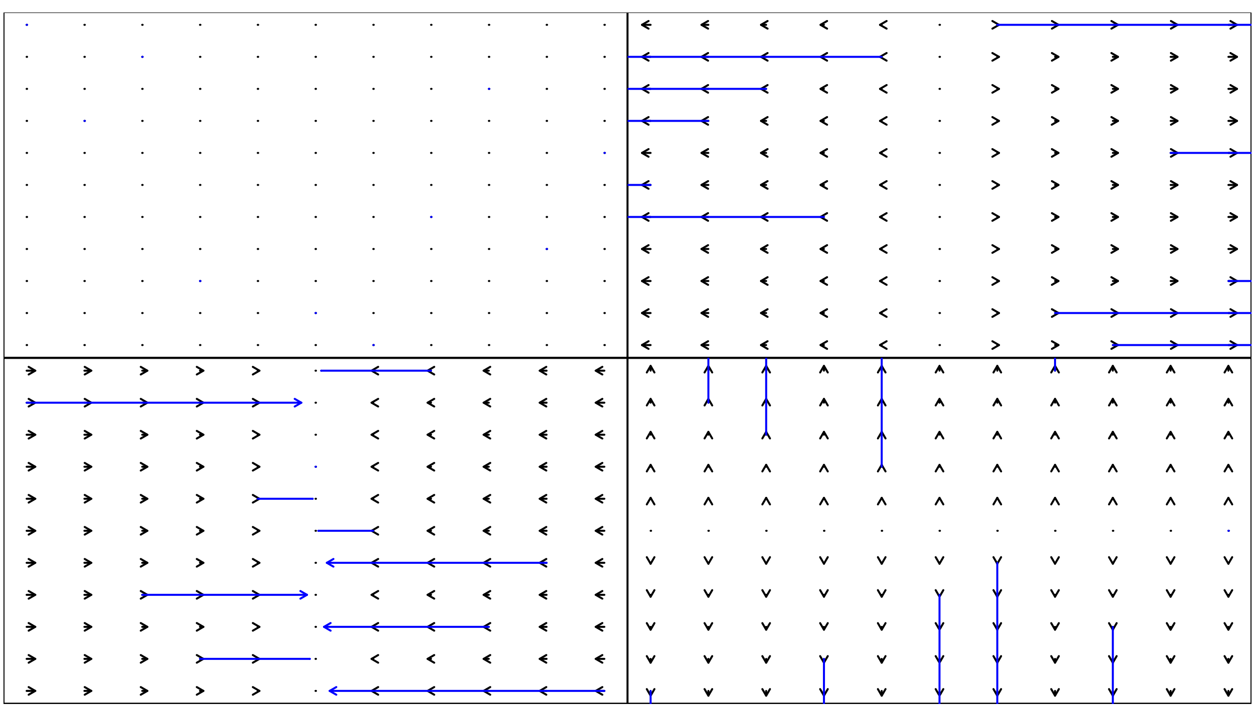
Vector fields

Robin Aldridge-Sutton

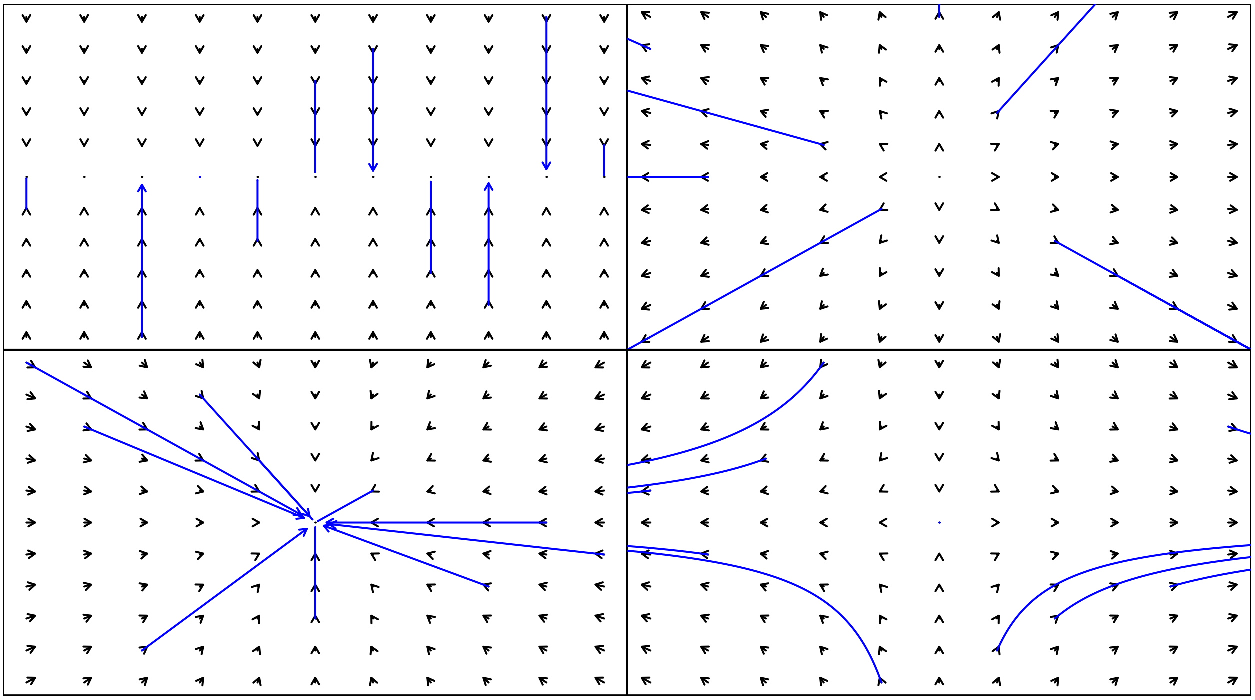
```r
# Function to plot vector fields and approximate solutions
plot_vf <- function(Fn) {
  # Mesh for plot
  l <- 11
  m <- seq(-2, 2, l = l)
  y <- rep(m, l)
  x <- rep(m, each = l)

  # Time step
  dt <- 0.03

  # Arrow head length
  arwl <- 0.04

  # Gradient over mesh
  Fxy <- Fn(x, y)

  # Plot vector field - gives warnings for points with zero gradient
  plot(x, y, type = 'n', axes = F)
  box()
  defaultW <- getOption("warn")
  options(warn = -1)
  arrows(x, y, x + dt * Fxy[, 1], y + dt * Fxy[, 2], length = arwl)
  options(warn = defaultW)

  # Number of time steps
  T <- 100

  # Number of solutions
  n_s <- 10

  # Randomly initialize solutions
  x_s <- y_s <- matrix(nrow = T, ncol = n_s)
  x_s[1, ] <- sample(m, n_s)
  y_s[1, ] <- sample(m, n_s)

  # Approximate solutions with Euler's method
  for (t in 2:T) {
    Fxys <- Fn(x_s[t - 1, ], y_s[t - 1, ])
    x_s[t, ] <- x_s[t - 1, ] + dt * Fxys[, 1]
    y_s[t, ] <- y_s[t - 1, ] + dt * Fxys[, 2]
  }

  # Plot solutions
```

```r
  matlines(x_s, y_s, col = 'blue', lty = 1)
  options(warn = -1)
  arrows(x_s[T - 1, ], y_s[T - 1, ], x_s[T, ], y_s[T, ], length = arwl,
         col = 'blue')
  options(warn = defaultW)
}

# Function to plot vector fields and approximate solutions for a linear
# system satisfying a certain set of eigenvalues
plot_vf_evals <- function(e_vals) {
  # Gradient function
  Fn <- function(x, y) {
    if (is.complex(e_vals)) {
      H <- 1 / sqrt(2) * cbind(c(1i, 1), c(1, 1i))
      Re(t(H %*% diag(e_vals) %*% t(Conj(H)) %*% rbind(x, y)))
    }
    else t(diag(e_vals) %*% rbind(x, y))
  }

  plot_vf(Fn)
}
```

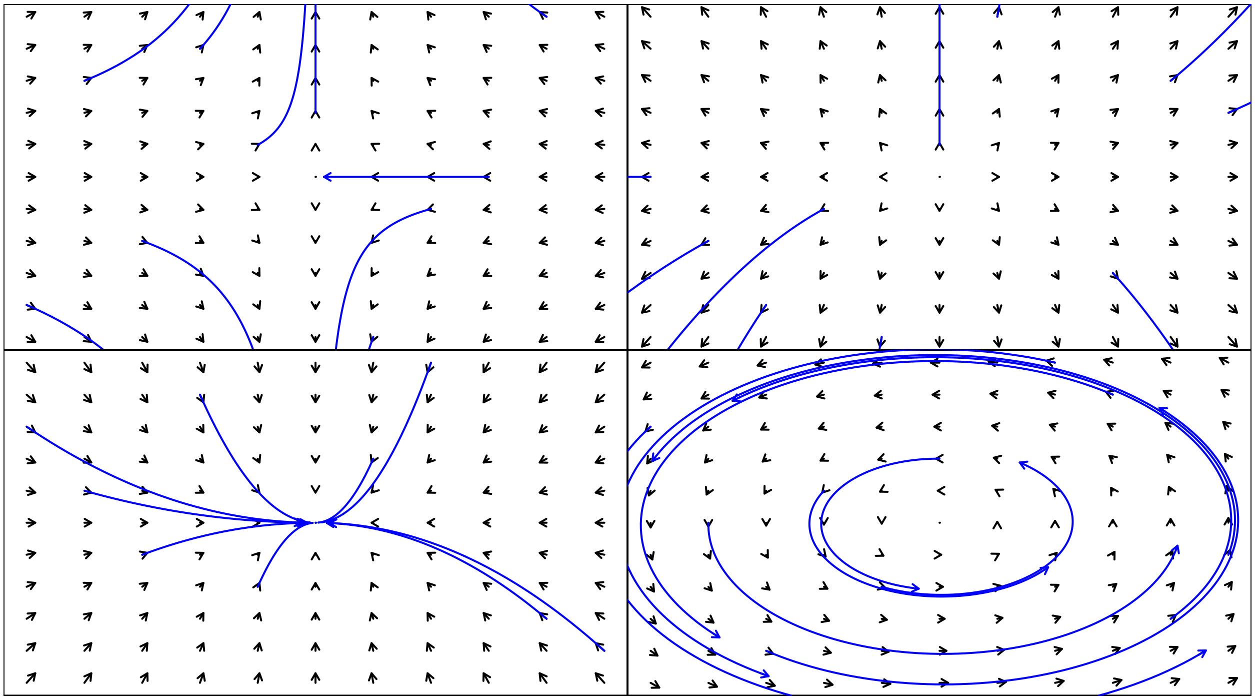
```r
# Plot vector fields and approximate solutions satisfying all  eigenvalues
par(mfrow = c(2, 2), mar = c(0, 0, 0, 0))
plot_vf_evals(c(0, 0))
plot_vf_evals(c(1, 0))
plot_vf_evals(c(-1, 0))
plot_vf_evals(c(0, 1))
```



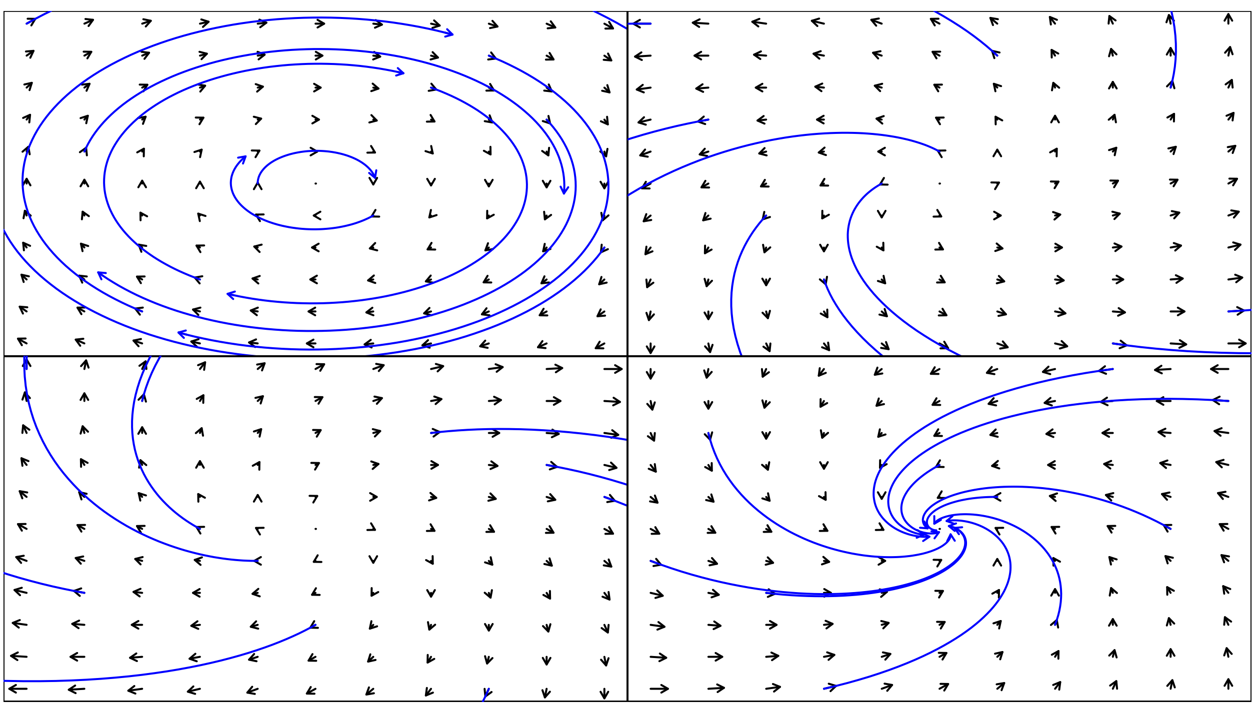
```
plot_vf_evals(c(0, -1))
plot_vf_evals(c(1, 1))
plot_vf_evals(c(-1, -1))
plot_vf_evals(c(1, -1))
```



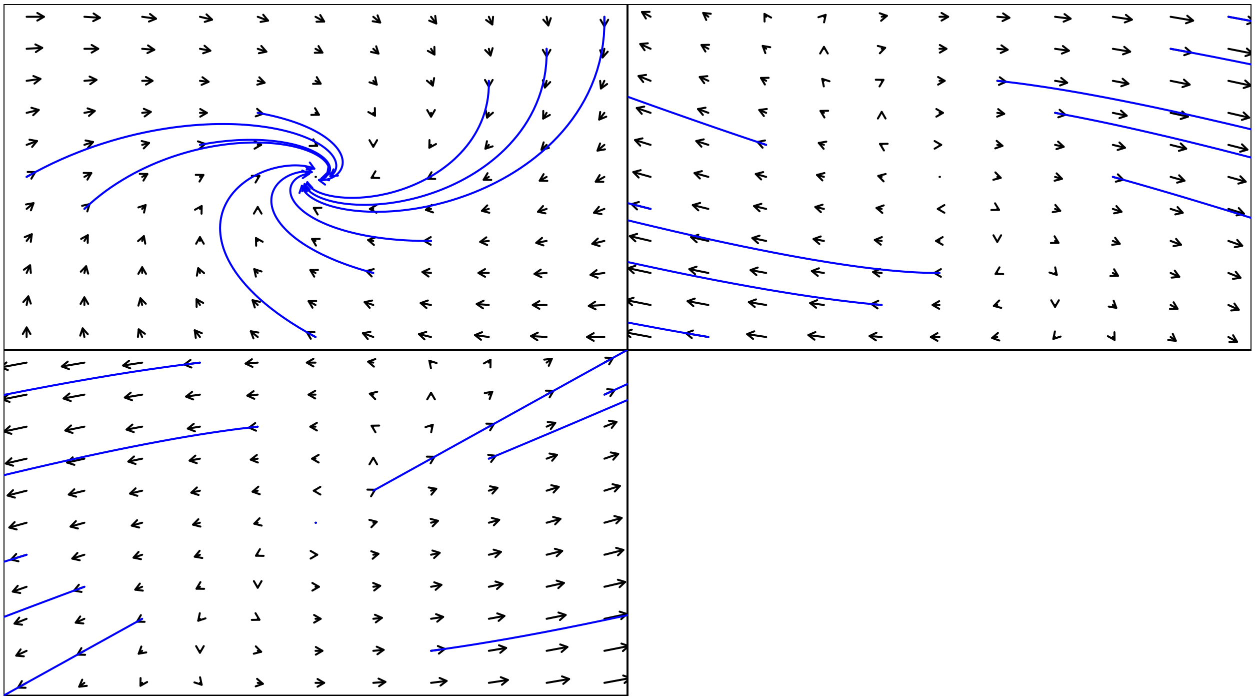
```
plot_vf_evals(c(-1, 1))
plot_vf_evals(c(1, 2))
plot_vf_evals(c(-1, -2))
plot_vf_evals(c(1i, -1i))
```



```
plot_vf_evals(c(-1i, 1i))
plot_vf_evals(c(1 + 1i, 1 - 1i))
plot_vf_evals(c(1 - 1i, 1 + 1i))
plot_vf_evals(c(-1 + 1i, -1 - 1i))
```

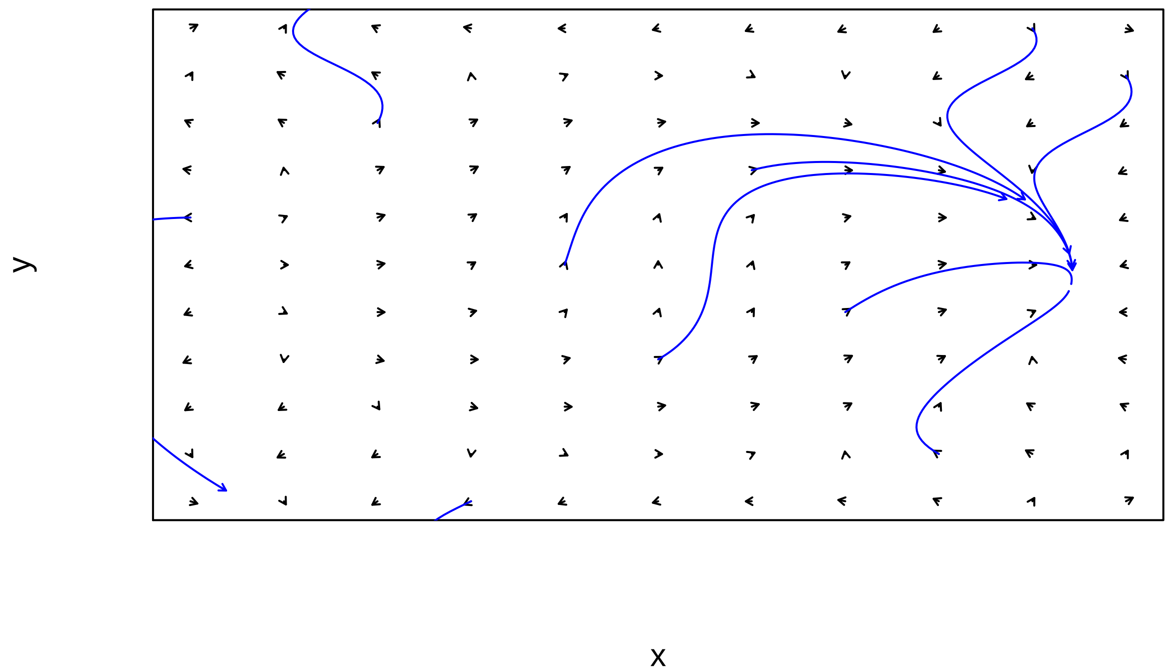


```
plot_vf_evals(c(-1 - 1i, -1 + 1i))
plot_vf(Fn = function(x, y) cbind(2 * x + y, -x))
plot_vf(Fn = function(x, y) cbind(2 * x - y, x))
```



The last case is a degenerate source, there is one repeated eigenvalue, and one linearly independent eigenvector, so there is no spectral decomposition. I'm not sure how to specify these matrices in general.

```
# Plot a nonlinear 2D system of differential equations
plot_vf(Fn = function(x, y) cbind(sin(x^2 + y^2), cos(x + y)))
```


y
x