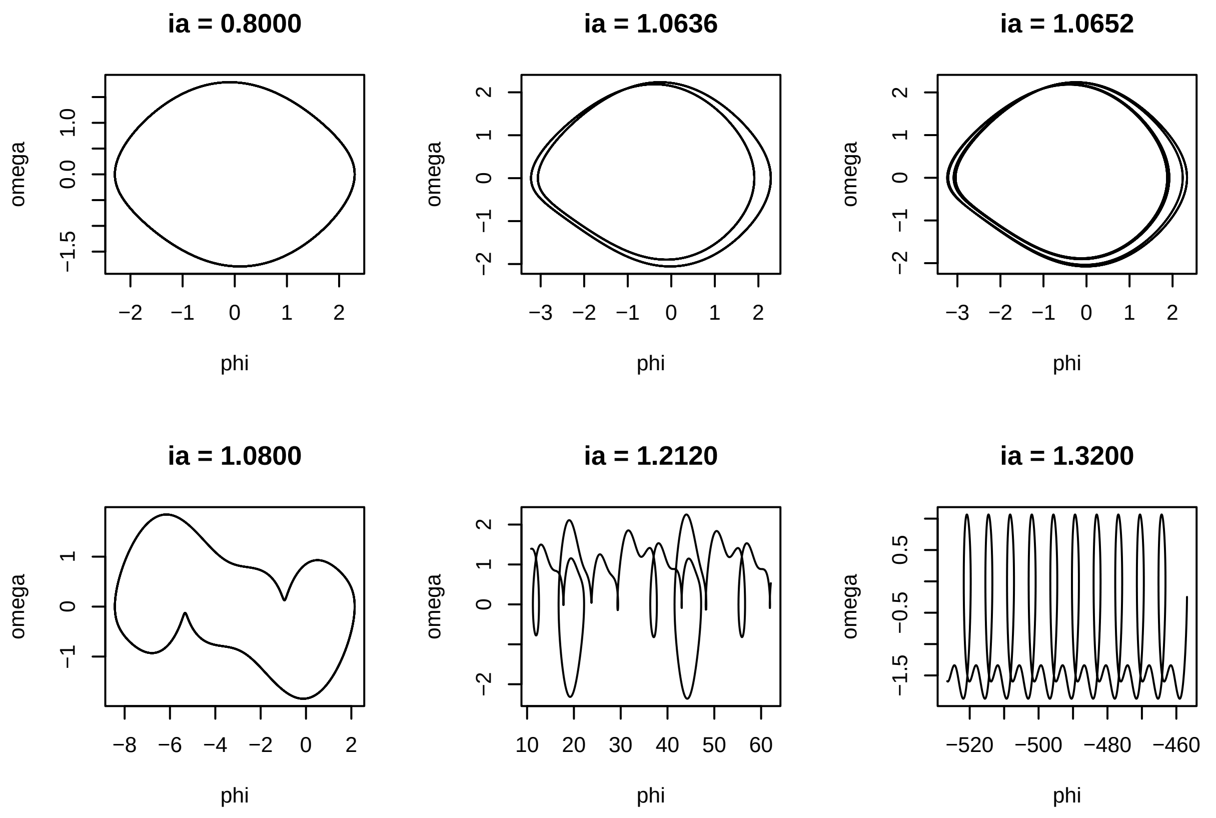
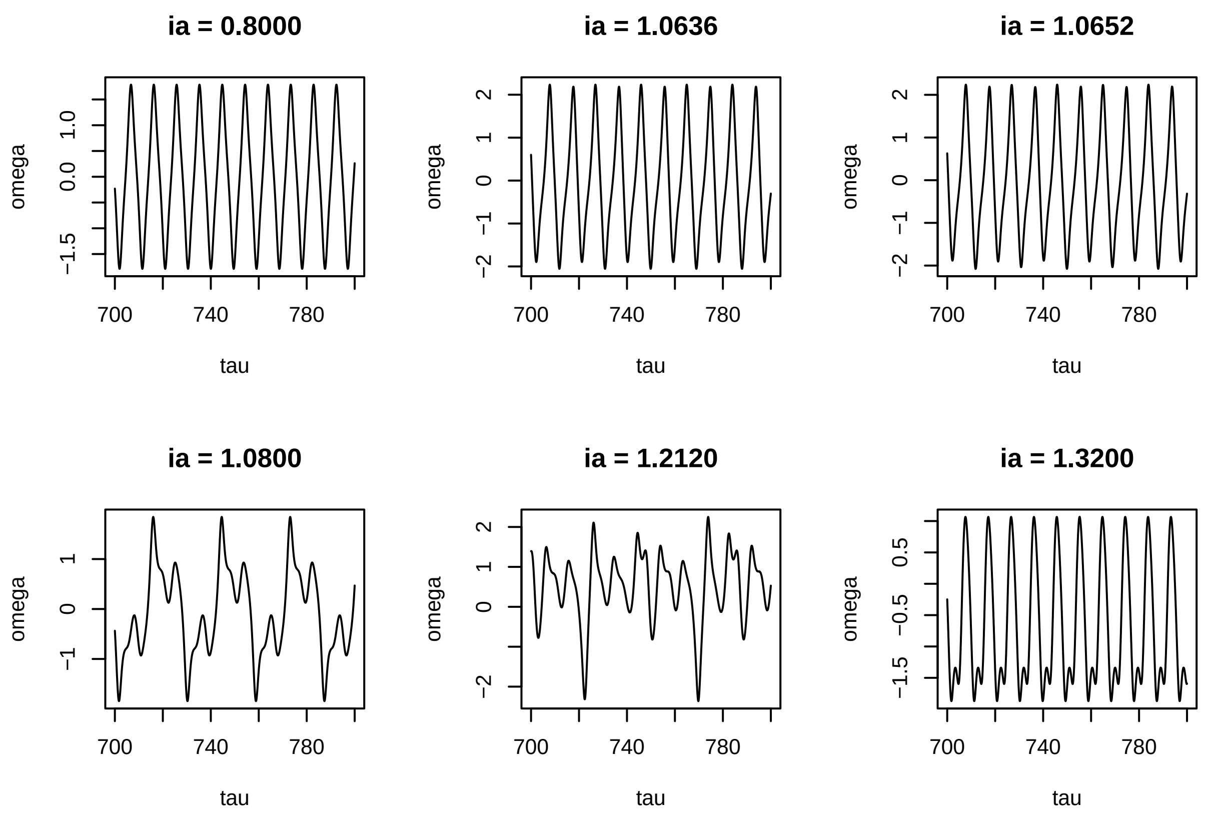
Josephson Simulation

以下所有的图像都是在 $\omega = 0.66$ ， $B_c = 0.5$ 下绘制的。

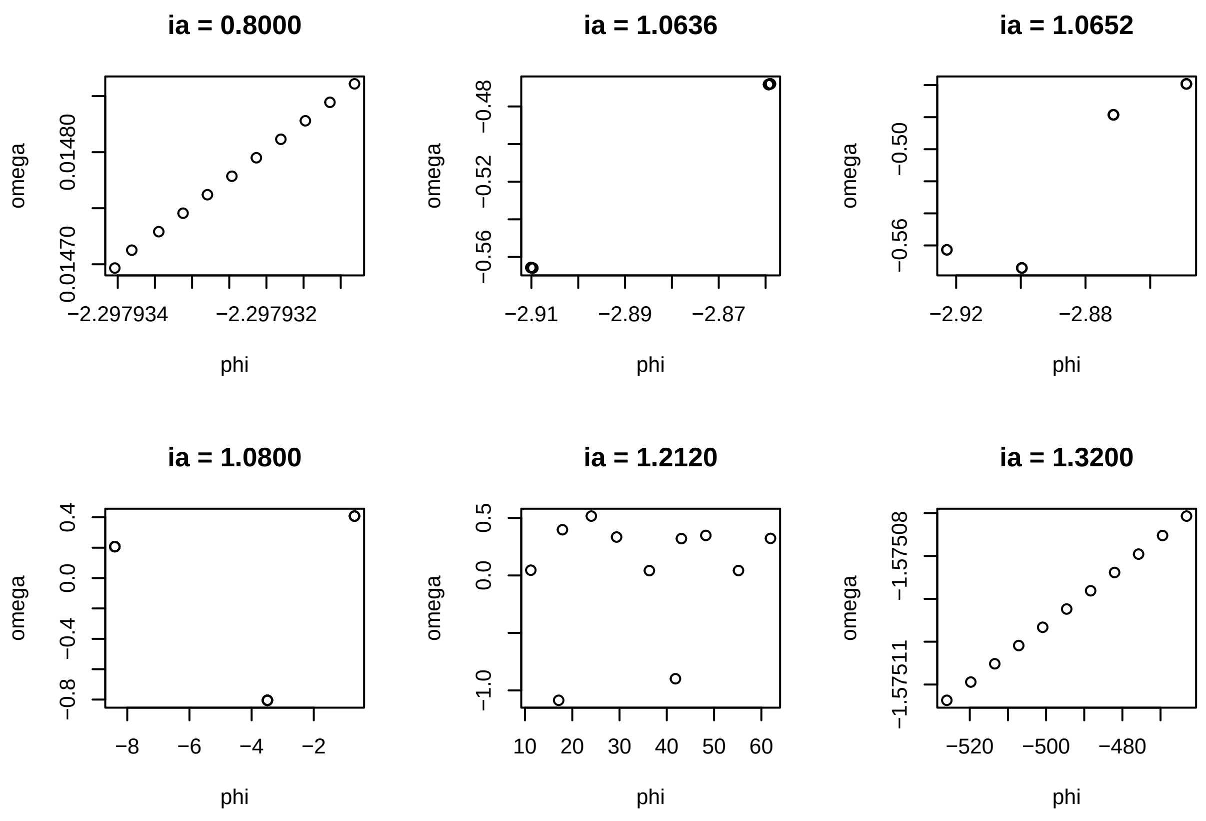
```r
par(mfrow=c(2,3))
ias = c("0.8000","1.0636", "1.0652", "1.0800", "1.2120", "1.3200")

for(ia in ias){
  ddd <- read.csv(paste0("chaos_",ia,".csv"))
  plot(ddd$phi, ddd$omega,
       main=paste0("ia = ", ia),
       xlab = "phi", ylab = "omega", type = "l")
}
```



```r
par(mfrow=c(2,3))
ias = c("0.8000","1.0636", "1.0652", "1.0800", "1.2120", "1.3200")

for(ia in ias){
  ddd <- read.csv(paste0("chaos_",ia,".csv"))

  plot(ddd$tau, ddd$omega,
       main=paste0("ia = ", ia),
       xlab = "tau", ylab = "omega", type = "l")
}
```

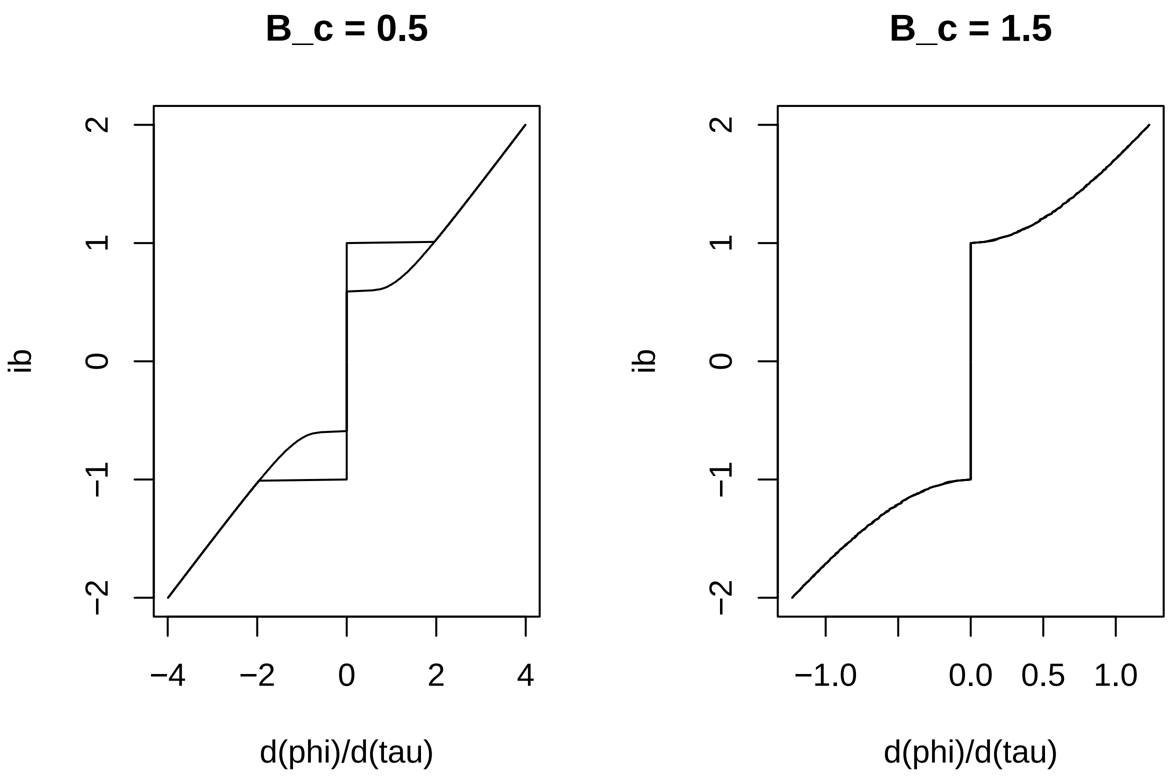


```
par(mfrow=c(2,3))
ias = c("0.8000","1.0636", "1.0652", "1.0800", "1.2120", "1.3200")

for(ia in ias){
  ddd <- read.csv(paste0("chaos_poincare_",ia,".csv"))

  plot(ddd$phi, ddd$omega,
       main=paste0("ia = ", ia),
       xlab = "phi", ylab = "omega", type = "p")
}
```



I-V 图。取 $B_c = 0.5$ 。

```
par(mfrow=c(1,2))
dd <- read.csv("iv_0.500.csv")
plot(dd$omega, dd$ib, type="l", xlab = "d(phi)/d(tau)", ylab = "ib", main = "B_c = 0.5")
dd <- read.csv("iv_1.500.csv")
plot(dd$omega, dd$ib, type="l", xlab = "d(phi)/d(tau)", ylab = "ib", main = "B_c = 1.5")
```



主要的模拟部分调用 GSL 库的 rk8pd 高精度算法：

```
int j_ode_system(double tau, const double * y, double * f, void * par) {
  j_ode_par * p = (j_ode_par *)par;

  double phi = y[0];
  double omega = y[1];

  double current = p->i_fun(tau, p->i_fun_context);

  f[0] = omega;
  f[1] = current - sin(phi) - p->B_c * omega;

  return GSL_SUCCESS;
}

j_ode_return run_josephson_ode(j_ode_par * jop, const sim_par *s) {
  double steps = s->steps;
  double tau_start = s->start;
  double tau_end = s->end;
  double h = (tau_end - tau_start) / steps;

  gsl_odeiv2_system sys = {j_ode_system, NULL, 2, jop};

  gsl_odeiv2_driver * driver = gsl_odeiv2_driver_alloc_y_new(&sys, gsl_odeiv2_step_rk8pd, ODE_HSTART, O

  double * phis = malloc(sizeof(double) * (steps + 1));
  double * omegas = malloc(sizeof(double) * (steps + 1));
```

```c
  double y[2] = {s->phi_0, s->omega_0};
  double tau = 0;
  j_ode_return r = {NULL, NULL};

  for(int i = 0; i <= steps; i++) {
    const double tau_i = i * h + tau_start;
    int status = gsl_odeiv2_driver_apply(driver, &tau, tau_i, y);
    if(status != GSL_SUCCESS) {
      fprintf(stderr, "SIMULATION ERROR\n");
      return r;
    }
    phis[i] = y[0];
    omegas[i] = y[1];
  }

  r.phis = phis;
  r.omegas = omegas;
  return r;

}
```