ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

ИССЛЕДОВАНИЕ РАБОТЫ ПИ-РЕГУЛЯТОРА

студент Пенкин Станислав

Вариант 17

18 февраля 2017

ИСХОДНЫЕ ДАННЫЕ:

```
V0 = 0        # начальная скорость
m = 100       # масса обьекта
dt = 1        # величина шага изменения времени (сек)
maxT = 400    # максимальное время достижения круизной скорости с точностью dV
dV = 0.1      # точность достижения скорости
```

ИСХОДНЫЕ ДАННЫЕ ВАРИАНТА:

```
kP = 2        # коэффициент пропорционального регулятора
kI = 0.2      # коэффициент интегрального регулятора
kc = 1.5      # коэффициент сопротивления среды
Vdir = 35     # круизная скорость
Umax = 100    # максимальное управляющее воздействие
```

ПРОГРАММА РАСЧЕТА ТРАЕКТОРИИ СКОРОСТИ (ПИ-РЕГУЛЯТОР):

```
cruizCtrl = function(kP, kI, kc, Vdir, Umax) {
  ind = 1:maxT
  Umin = (-Umax)
  T = ind * dt
  v = T * 0
  err = 0
  sumerr = 0
  v[1] = V0
  for (i in ind[-maxT]) {
    err = Vdir - v[i]
    sumerr = sumerr + err
    u = kP * err + kI * sumerr * dt
    if (u > Umax) u = Umax
    if (u < Umin) u = Umin
    v[i + 1] = v[i] + (u - kc * v[i]) * dt / m
  }
## необходимо вернуть data.frame - в 1-м столбце - вектор времен, во 2-м - вектор скоростей
  data.frame(T, v)
}
```

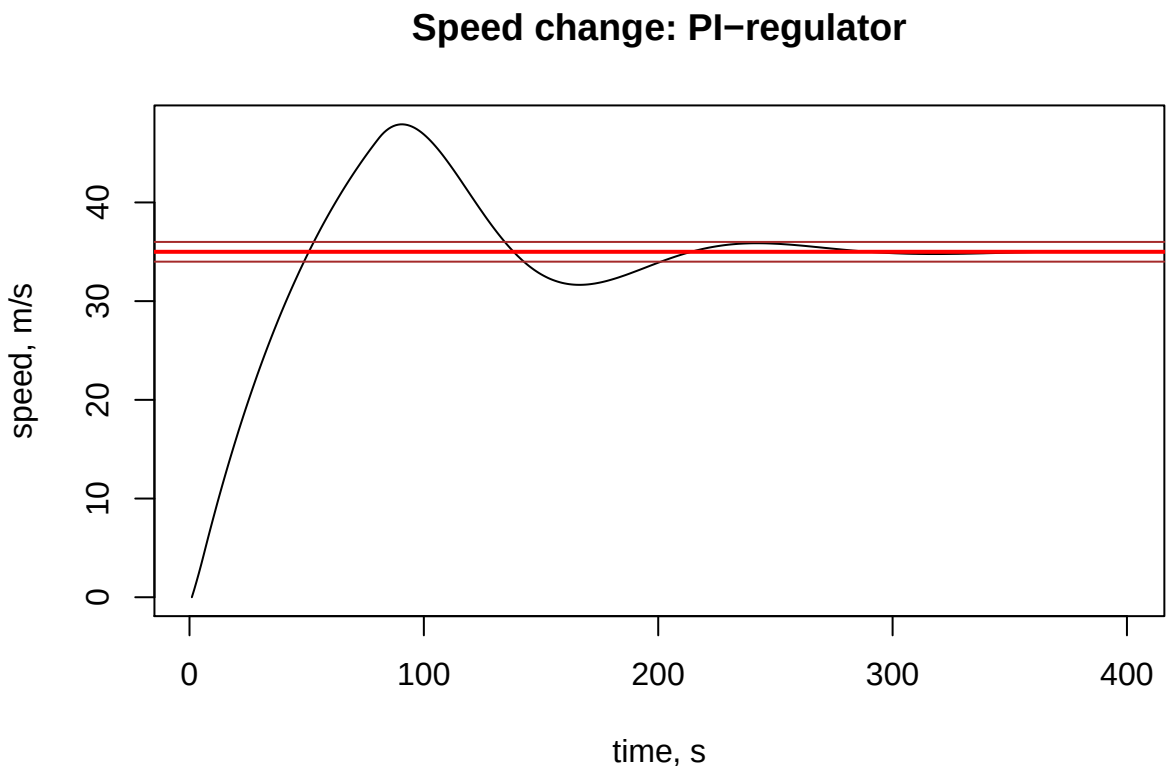
1. АНАЛИЗ РАБОТЫ ПИ-РЕГУЛЯТОРА ПРИ ЗАДАННЫХ ЗНАЧЕНИЯХ.

РАСЧЕТ ТРАЕКТОРИИ СКОРОСТИ ПРИ ЗАДАННЫХ ЗНАЧЕНИЯХ ПАРАМЕТРОВ ПИ-РЕГУЛЯТОРА:

```
res = cruizCtrl(kP, kI, kc, Vdir, Umax)
head(res)
```

```
##   T        v
## 1 1 0.000000
## 2 2 0.770000
## 3 3 1.581510
## 4 4 2.431454
## 5 5 3.316787
## 6 6 4.234500
```

ВЫВОД ГРАФИКА:



ВЫВОД №1:

При заданных значениях регулятора kP=2 , kl=0.2 система достигает заданой точности за время, меньшее, чем maxT = 400 сек.

2. ИССЛЕДУЕМ ПИ-РЕГУЛЯТОР.

ПРОГРАММА РАСЧЕТА ВРЕМЕНИ ДОСТИЖЕНИЯ ЗАДАННОЙ ТОЧНОСТИ ПО СКОРОСТИ:

```
getTime = function(tv, Vdir, dV) {
  T = tv[["T"]]
```

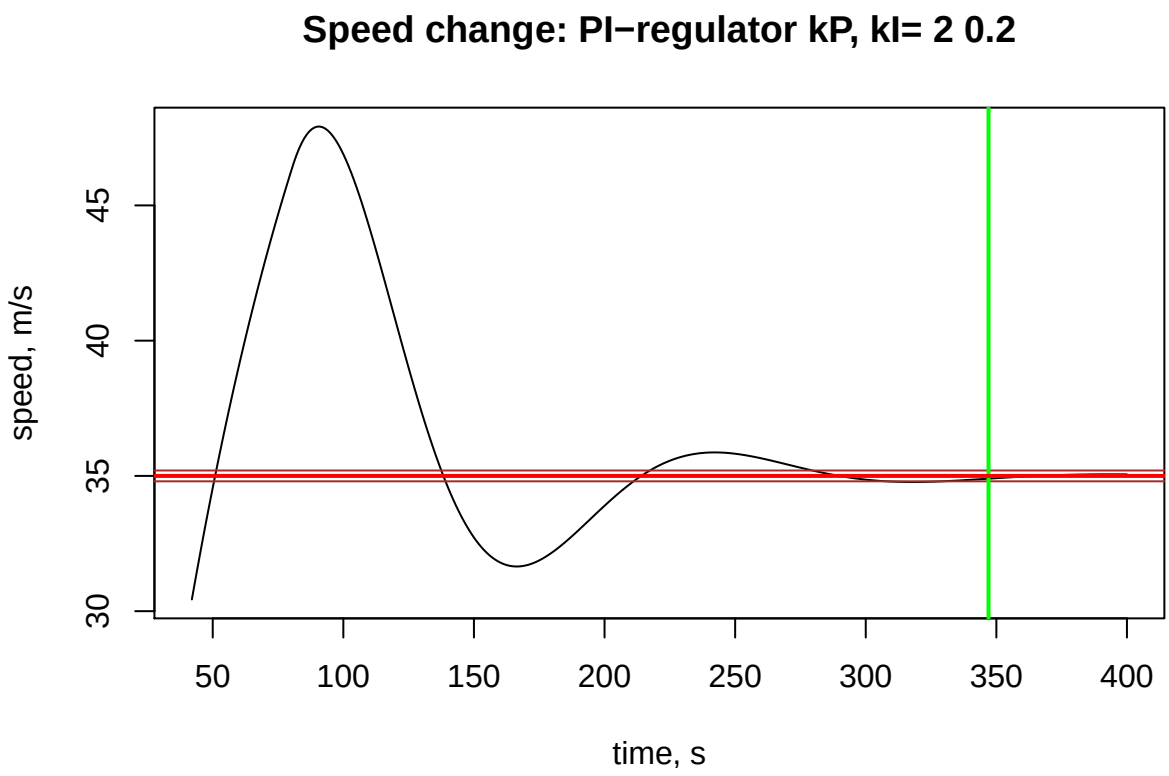
```
  v = tv[["v"]]
  i = length(v)
  while ((abs(Vdir - v[i]) < dV) & (i > 1)) {
    i = i - 1
  }
## необходимо вернуть время, начиная с которого скорость отклонялась от заданной
## меньше, чем на dV
  T[i]
}
```

РАСЧЕТ ВРЕМЕНИ ДОСТИЖЕНИЯ ЗАДАННОЙ ТОЧНОСТИ ПРИ ЗАДАННЫХ ЗНАЧЕНИЯХ ПАРАМЕТРОВ ПИ-РЕГУЛЯТОРА:

```
tLim = getTime(res, Vdir, dV)
print(paste(tLim, Vdir, dV))
```

```
## [1] "347 35 0.1"
```

ВЫВОД ГРАФИКА:



РАСЧЕТ ТРАЕКТОРИИ СКОРОСТИ ПРИ ВЫБРАННЫХ ЗНАЧЕНИЯХ ПАРАМЕТРОВ ПИ-РЕГУЛЯТОРА:

```
res1 = cruizCtrl(2, 0.1, kc, Vdir, Umax)
tail(res1)
```

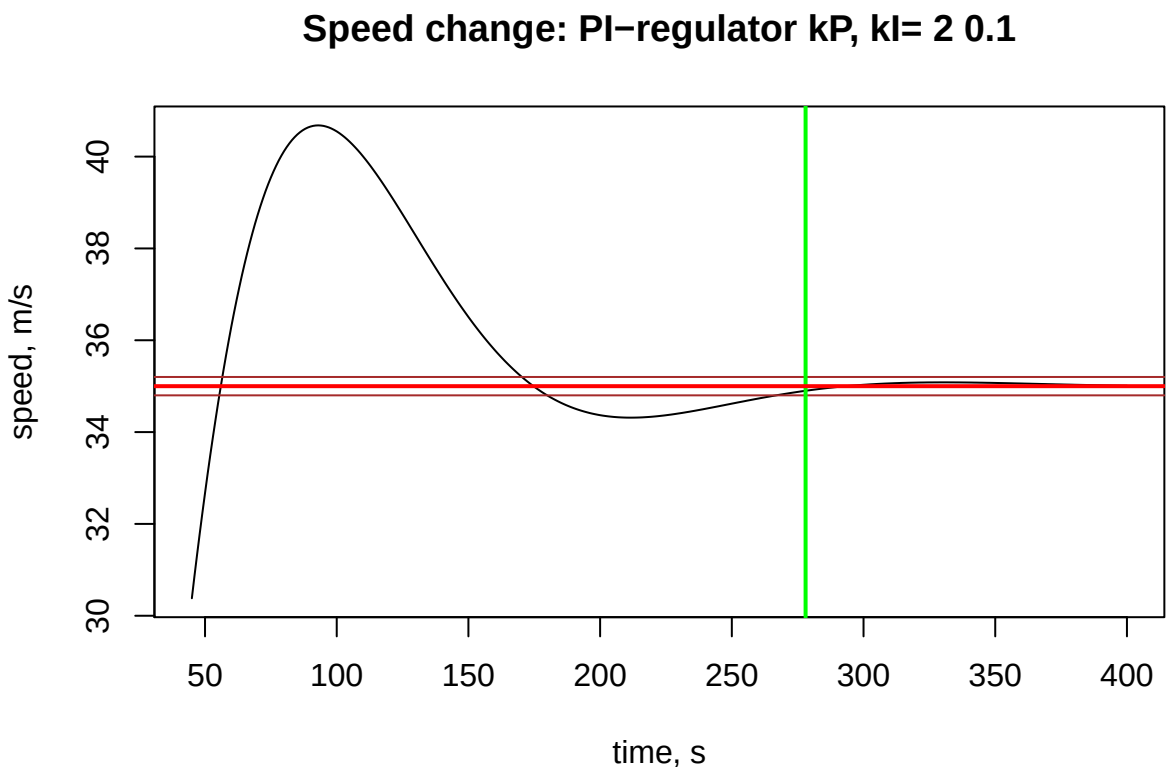
```
##       T       v
## 395 395 35.01376
## 396 396 35.01277
## 397 397 35.01181
## 398 398 35.01087
## 399 399 35.00995
## 400 400 35.00905
```

РАСЧЕТ ВРЕМЕНИ ДОСТИЖЕНИЯ ЗАДАННОЙ ТОЧНОСТИ ПРИ ВЫБРАННЫХ ЗНАЧЕНИЯХ ПАРАМЕТРОВ ПИ-РЕГУЛЯТОРА:

```
tLim1 = getTime(res1, Vdir, dV)
print(paste(tLim1, Vdir, dV))
```

```
## [1] "278 35 0.1"
```

ВЫВОД ГРАФИКА:



ВЫВОД №2:

При выбраных значениях ПИ-регулятора kP=2, kI=0.1 система достигает заданной точности в момент времени t1=278 сек. Это быстрее, чем момент времени t=347 сек при заданных значениях ПИ-регулятора.

3. ИССЛЕДУЕМ ПОРЯДОК И СКОРОСТЬ СХОДИМОСТИ.

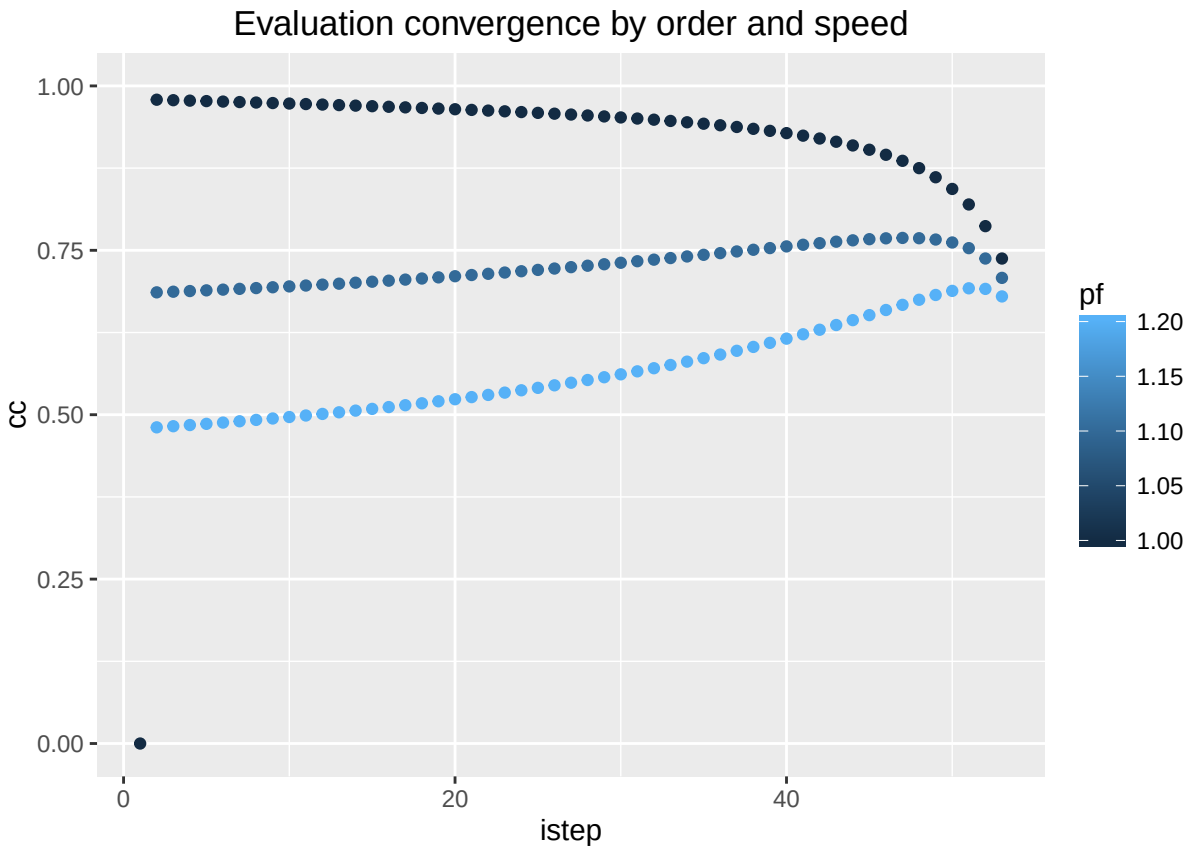
ПРОГРАММА РАСЧЕТА СТАБИЛИЗИРУЕМОЙ ПОСЛЕДОВАТЕЛЬНОСТИ:

```
getCC = function(v, Vdir, p) {
  v = v[v < Vdir - 0.1]
  numv = length(v)
  cc = v * 0
  istep = rep(1:numv, length(p))
  pf = rep(p, each = numv)
  vf = rep(v, length(p))
  for (i in 2:length(vf)) {
    cc[i] = abs(Vdir - vf[i]) / abs(Vdir - vf[i - 1]) ^ pf[i]
  }
## необходимо вернуть data.frame - в 1-м столбце - вектор номеров шагов,
## во 2-м - стабилизируемая последовательность,
## в 3-м - фактор разных споледовательностей (при разных значениях p)
  data.frame(istep, cc, pf)
}
```

ОЦЕНКА ПОРЯДКА И СКОРОСТИ СХОДИМОСТИ:

```
res2 = res1[res1$v < Vdir - dV * 10,]
ccframe = getCC(res2$v, Vdir, c(1, 1.1, 1.2))
```

ВЫВОД ГРАФИКА



Видим, что стабилизация последовательности достигается при порядке p≈ 1.1, поэтому выбираем порядок сходимости равным 1.1.

```
order = 1.1
ccframe1 = getCC(res2$v, Vdir, c(order))
velocity = 1 / mean(ccframe1$cc)
print(velocity)
```

```
## [1] 1.403866
```

ВЫВОД №3:

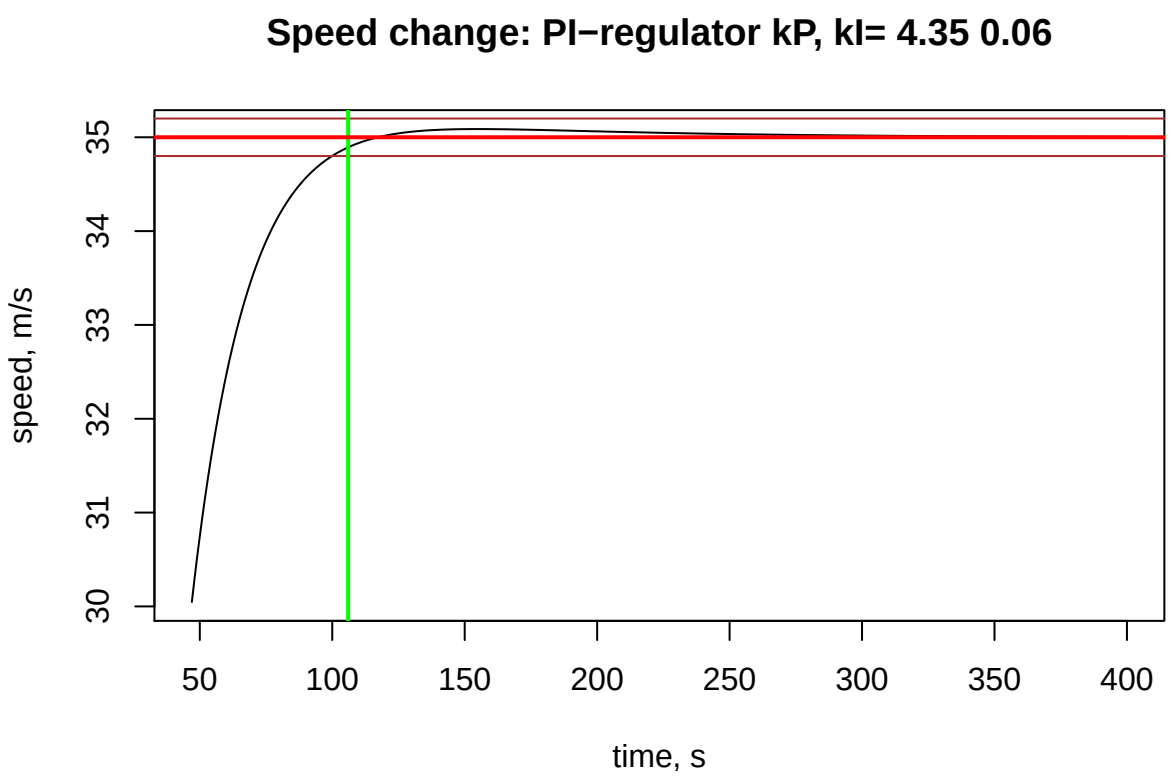
При выбраных значениях kP=2, kI=0.1 ПИ-регулятора система сходится с порядком сходимости p=1.1 и со скоростью сходимости velocity=1.403866.

4. ИССЛЕДУЕМ ОПТИМАЛЬНЫЕ ПАРАМЕТРЫ РЕГУЛЯТОРА.

```
kP = c(seq(from = 0, to = 5, by = 0.05))
kI = c(seq(from = 0, to = 1, by = 0.01))
for (iI in 1:101) {
  for (iP in 1:101) {
    res = cruizCtrl(kP[iP], kI[iI], kc, Vdir, Umax)
    tLim = getTime(res, Vdir, dV)
    data[i, 1] = kP[iP]
    data[i, 2] = kI[iI]
```

```
    data[i, 3] = tLim
    i = i + 1
  }
}
opt = data[which.min(data$tLim),]
res = cruizCtrl(opt$kP, opt$kI, kc, Vdir, Umax)
```

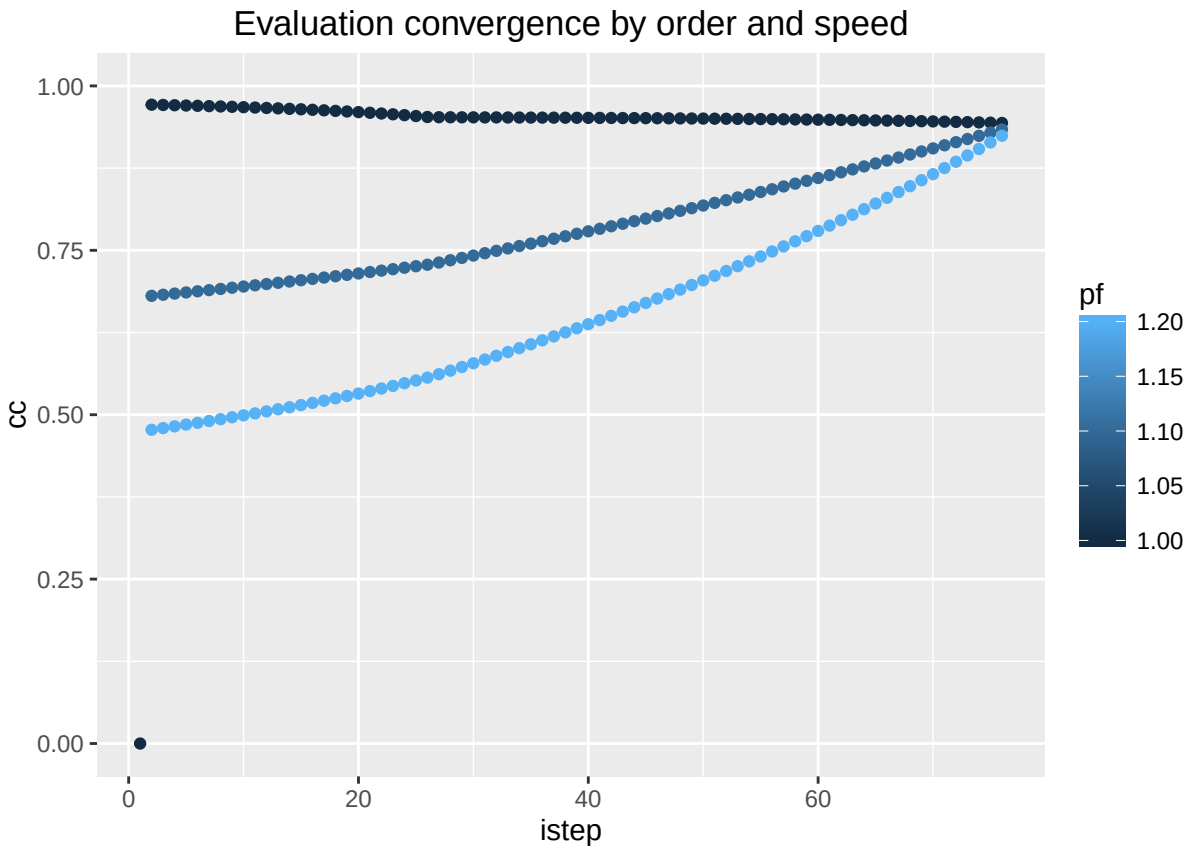
ВЫВОД ГРАФИКА



ОЦЕНКА ПОРЯДКА И СКОРОСТИ СХОДИМОСТИ:

```
res1 = res[res$v < Vdir - dV * 10,]
ccframe = getCC(res1$v, Vdir, c(1, 1.1, 1.2))
```

ВЫВОД ГРАФИКА



Видим, что стабилизация последовательности достигается при порядке p≈ 1.01, поэтому выбираем порядок сходимости равным 1.01.

```
order = 1.01
ccframe1 = getCC(res2$v, Vdir, c(order))
velocity = 1 / mean(ccframe1$cc)
print(velocity)
```

```
## [1] 1.115976
```

ВЫВОД №4:

Найдены лучшие параметры ПИ-регулятора kP=4.35 , kI=0.06 . Время достижения заданной точности управления составило t=106 сек . Скорость сходимости равна velocity=1.115976 при порядке сходимости p=1.01.

Регулировка kP отвечает за изменение воздействия в зависимости от разницы крейсерской и текущей скорости. Чем больше kP, тем большее воздействие будет на обьект. Регулировка kI нужна для компенсации возрастающего с увеличением скорости сопротивления среды. Значение kI надо выбирать исходя из того, насколько велико сопротивление среды.