```
N := 20:
f1 := (x, y, u) → u:
f2 := (x, y, u) →  -y+x:
x0 := 0:
 u0 := 2.19:
y0 := 0:
xf := 1:
h := evalf((xf - x0) / (N-1)): # step size
 m := 1:       # to print every mth step
 fmt := `%10.4f %10.4f %10.4f \n`:
x := x0:   y := y0:   u := u0:

for i from 1 to N - 1 do
  k1 := f1(x, y, u):
  l1 := f2(x, y, u):            # left-hand slopes
  k2 := f1(x + h / 2, y + h* k1 / 2, u + h* l1 / 2):
  l2 := f2(x + h / 2, y + h* k1 / 2, u + h* l1 / 2): # 1st midpt slopes
  k3 := f1(x + h / 2, y + h* k2 / 2, u + h* l2 / 2):
  l3 := f2(x + h / 2, y + h* k2 / 2, u + h* l2 / 2): # 2nd midpt slopes
  k4 := f1(x + h, y + h* k3, u + h* l3):
  l4 := f2(x + h, y + h* k3, u + h* l3):    # right-hand slopes
  k := (k1 + 2 * k2 + 2 * k3 + k4) / 6:  # average x-slope
  l := (l1 + 2 * l2 + 2 * l3 + l4) / 6:  # average y-slope
  y := y + h* k:     # update x
  u := u + h* l:     # update y
  x := x + h:       # update t
  if trunc(i / m) = i / m then       # display each
    printf(fmt, x, y, u) fi; #  mth value
  od:
```

```
0.0526     0.1152     2.1884
0.1053     0.2303     2.1834
0.1579     0.3450     2.1752
0.2105     0.4592     2.1637
0.2632     0.5727     2.1490
0.3158     0.6854     2.1312
0.3684     0.7970     2.1101
0.4211     0.9074     2.0861
0.4737     1.0165     2.0590
0.5263     1.1241     2.0289
0.5789     1.2300     1.9961
0.6316     1.3342     1.9604
0.6842     1.4364     1.9222
0.7368     1.5365     1.8813
0.7895     1.6344     1.8380
0.8421     1.7299     1.7924
0.8947     1.8230     1.7446
0.9474     1.9135     1.6947
```

1.0000 2.0014 1.6430