A greater population of white daisies will tend to increase planetary albedo and decrease the emission temperature, as is apparent from equation ([lab5:eq:tempe]), while the reverse is true for the black daisies.

To summarize: The daisy population is controlled by its growth rate β_i which is a function of the local temperature T_i $$\beta_i = 1.0 - 0.003265(295.5 K -T_i)^2$$ If the conductivity R is nonzero, the local temperature is a function of planetary albedo α_p

$$T_i = \left[R L \frac{S_0}{4\sigma}(\alpha_p-\alpha_i) + T^4_e \right]^{\frac{1}{4}}$$

which is determined by the daisy population.

- Physically, this provides the feedback from the daisy population back to the temperature, completing the loop between the daisies and temperature.
- Mathematically, this introduces a rather nasty non-linearity into the equations which, as pointed out in the lab 1, usually makes it difficult, if not impossible, to obtain exact analytic solutions.

Problem Initial

The feedback means a stable daisy population (a steady state) and the environmental conditions are in a delicate balance. The code below produces a steady state which arises from a given initial daisy population that starts with only white daisies.

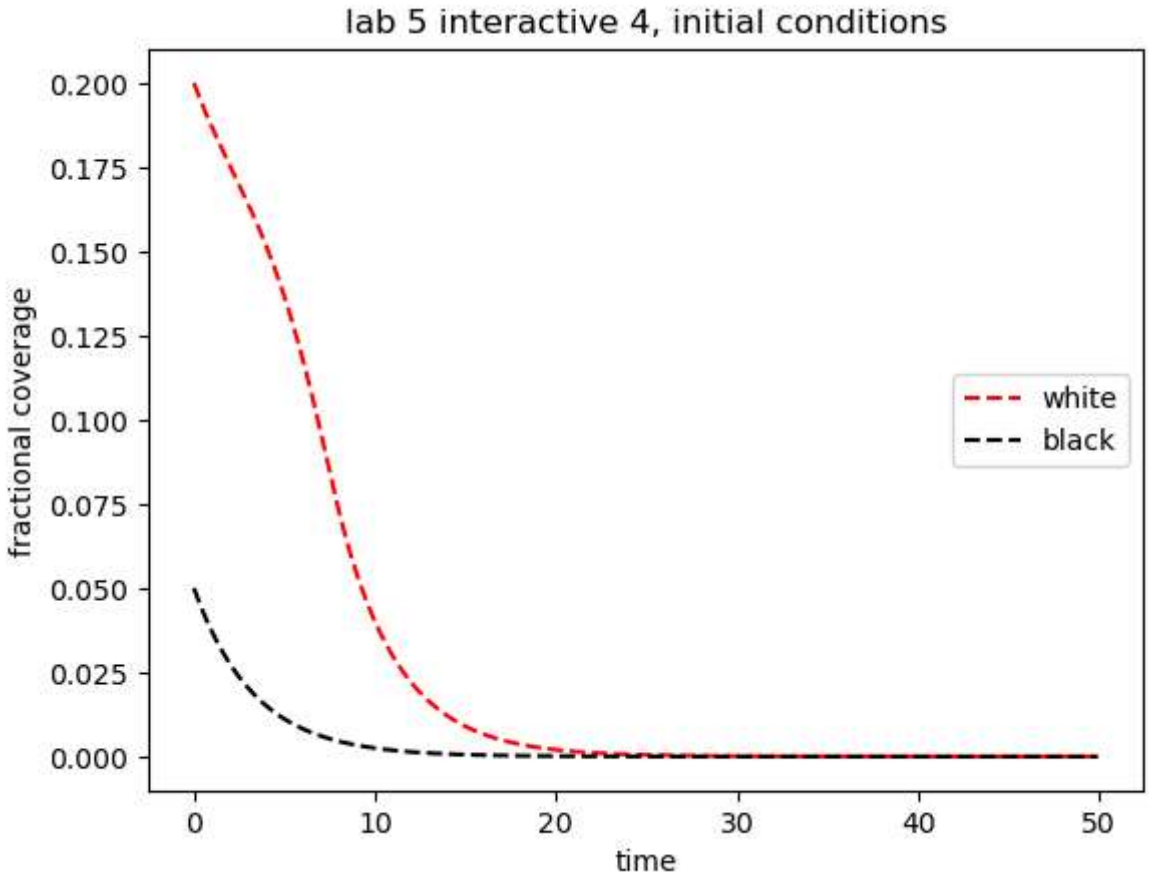
1. Add a relatively small (5%, blackconc = 0.05) initial fraction of black daisies to the value in initial.yaml and see what effect this has on the temperature and final daisy populations. Do you still have a final non-zero daisy population?
2. Set the initial black daisy population to 0.05 Attempt to adjust the initial white daisy population to obtain a non-zero steady state. What value of initial white daisy population gives you a non-zero steady state for blackconc=0.05? Do you have to increase or decrease the initial fraction? What is your explanation for this behavior?
3. Experiment with other initial fractions of daisies and look for non-zero steady states. Describe and explain your results. Connect what you see here with the discussion of hysteresis towards the end of this lab - what determines which steady state is reached?

In [36]:
```
# functions for problem initial
from numlabs.lab5.lab5_funs import Integrator


class Integ54(Integrator):
```

```
    def set_yinit(self):
        #
        # read in 'albedo_white chi S0 L albedo_black R albedo_ground'
        #
        uservars = namedtuple('uservars', self.config['uservars'].keys())
        self.uservars = uservars(**self.config['uservars'])
        #
        # read in 'whiteconc blackconc'
        #
        initvars = namedtuple('initvars', self.config['initvars'].keys())
        self.initvars = initvars(**self.config['initvars'])
        self.yinit = np.array(
            [self.initvars.whiteconc, self.initvars.blackconc])
        self.nvars = len(self.yinit)
        return None

    def __init__(self, coeff_file_name):
        super().__init__(coeff_file_name)
        self.set_yinit()

    def find_temp(self, yvals):
        """
            Calculate the temperatures over the white and black daisies
            and the planetary equilibrium temperature given the daisy fractions

            input:  yvals -- array of dimension [2] with the white [0] and black [
                    daisy fractiion
            output:  white temperature (K), black temperature (K), equilibrium tem
        """
        sigma = 5.67e-8  # Stefan Boltzman constant W/m^2/K^4
        user = self.uservars
        bare = 1.0 - yvals[0] - yvals[1]
        albedo_p = bare * user.albedo_ground + \
            yvals[0] * user.albedo_white + yvals[1] * user.albedo_black
        Te_4 = user.S0 / 4.0 * user.L * (1.0 - albedo_p) / sigma
        temp_e = Te_4**0.25
        eta = user.R * user.L * user.S0 / (4.0 * sigma)
        temp_b = (eta * (albedo_p - user.albedo_black) + Te_4)**0.25
        temp_w = (eta * (albedo_p - user.albedo_white) + Te_4)**0.25
        return (temp_w, temp_b, temp_e)

    def derivs5(self, y, t):
        """y[0]=fraction white daisies
           y[1]=fraction black daisies
           no feedback between daisies and
           albedo_p (set to ground albedo)
        """
        temp_w, temp_b, temp_e = self.find_temp(y)

        if (temp_b >= 277.5 and temp_b <= 312.5):
            beta_b = 1.0 - 0.003265 * (295.0 - temp_b)**2.0
        else:
            beta_b = 0.0

        if (temp_w >= 277.5 and temp_w <= 312.5):
            beta_w = 1.0 - 0.003265 * (295.0 - temp_w)**2.0
```

```
        else:
            beta_w = 0.0
        user = self.uservars
        bare = 1.0 - y[0] - y[1]
        # create a 1 x 2 element vector to hold the derivitive
        f = np.empty_like(y)
        f[0] = y[0] * (beta_w * bare - user.chi)
        f[1] = y[1] * (beta_b * bare - user.chi)
        return f
```

In [9]:
```
# Solve and plot for problem initial
import matplotlib.pyplot as plt
import pandas as pd

theSolver = Integ54('initial.yaml')
timevals, yvals, errorlist = theSolver.timeloop5fixed()
daisies = pd.DataFrame(yvals, columns=['white', 'black'])

thefig, theAx = plt.subplots(1, 1)
line1, = theAx.plot(timevals, daisies['white'])
line2, = theAx.plot(timevals, daisies['black'])
line1.set(linestyle='--', color='r', label='white')
line2.set(linestyle='--', color='k', label='black')
theAx.set_title('lab 5 interactive 4, initial conditions')
theAx.set_xlabel('time')
theAx.set_ylabel('fractional coverage')
out = theAx.legend(loc='center right')
```

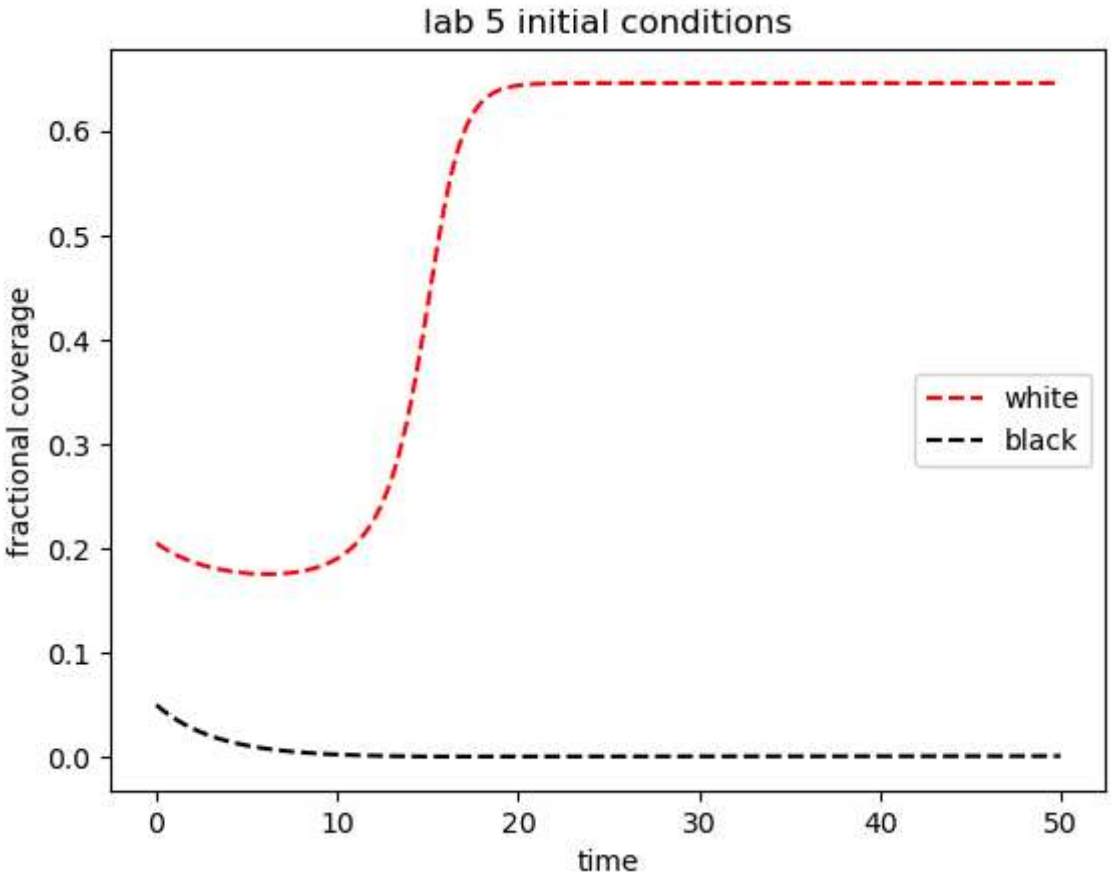


1. The final daisy population for both white and black is now zero with an initial condition of black daisies of 0.05. The temperatures also drastically increase until they stabilize at 324 K because of this initial daisy fraction

2. If we increase the initial condition for the white daisies to be 0.205, the white daisy population will attain a non-zero steady state solution, anything under that value will lead to extinction for the white daisies, so we need to increase the initial condition. In addition, it seems that no matter what value of initial condition is put for the white daisies, the black fraction will end up at zero as the steady state.

It seems that initially the emission temperature is too high for both daisy populations, but eventually when the black daisies go extinct, then the white daisies now have just the correct amount to shift the system out of being too warm to the correct temperature in order for them to grow and stabilize the temperature at a reasonable amount in order for them to survive (i.e. steady state). Anything more and the population would go extinct.

In [37]:
```
theSolver = Integ54('initial_V2.yaml')
timevals, yvals, errorlist = theSolver.timeloop5fixed()
daisies = pd.DataFrame(yvals, columns=['white', 'black'])

thefig, theAx = plt.subplots(1, 1)
line1, = theAx.plot(timevals, daisies['white'])
line2, = theAx.plot(timevals, daisies['black'])
line1.set(linestyle='--', color='r', label='white')
line2.set(linestyle='--', color='k', label='black')
theAx.set_title('lab 5 initial conditions')
theAx.set_xlabel('time')
theAx.set_ylabel('fractional coverage')
out = theAx.legend(loc='center right')
```

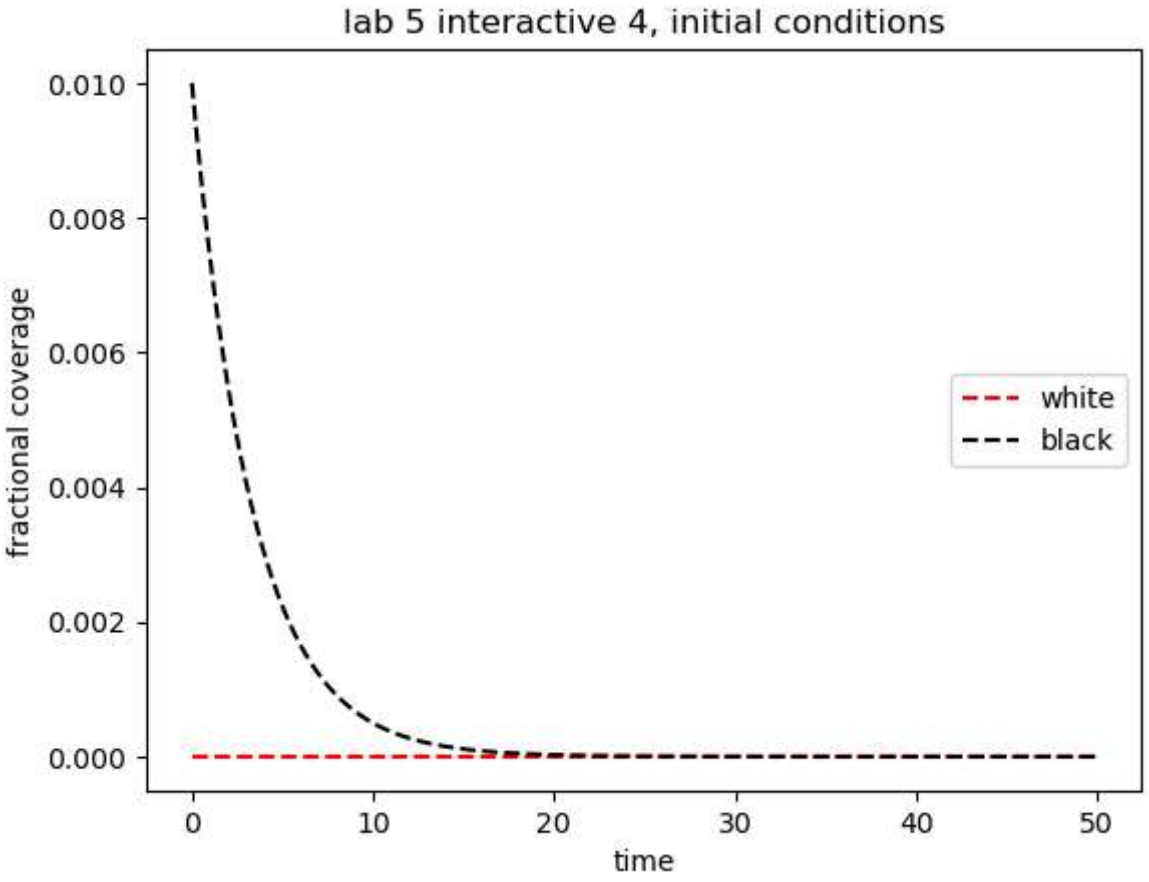


3. It seems that no matter what the value for the initial fraction of black daisies (regardless of the initial white daisy value as well), the black daisies will eventually go extinct, likely because the temperatures are just too warm for them to exist in any case. As for the white daisies, they only persist if their initial fraction is some degree larger than the initial black daisies. So, if the initial conditions are equal, or if the white daisies are less than the black daisies, both populations will die. In addition, if there are zero black daisies initially, the white daisies will only persist if their initial fraction is larger than 0.1, since otherwise they won't be able to influence the system enough to drop the temperatures to an equilibrium value.

In connection with the discusion at the end, we very much see the expected behavior of the black daisies, which always seem to go to zero with the solar constant with which we are working with. In addition, we see the behavior that white actually achieves a non-zero steady state solution when the initial fraction is increased.

In [11]:
```
theSolver = Integ54('initial_V3.yaml')
timevals, yvals, errorlist = theSolver.timeloop5fixed()
daisies = pd.DataFrame(yvals, columns=['white', 'black'])

thefig, theAx = plt.subplots(1, 1)
line1, = theAx.plot(timevals, daisies['white'])
line2, = theAx.plot(timevals, daisies['black'])
line1.set(linestyle='--', color='r', label='white')
```

```
line2.set(linestyle='--', color='k', label='black')
theAx.set_title('lab 5 interactive 4, initial conditions')
theAx.set_xlabel('time')
theAx.set_ylabel('fractional coverage')
out = theAx.legend(loc='center right')
```



Problem Temperature

The code above in Problem Initial adds a new method, `find_temp` that takes the white/black daisy fractions and calculates local and planetary temperatures.

1. override `timeloop5fixed` so that it saves these three temperatures, plus the daisy growth rates to new variables in the Integ54 instance
2. Make plots of (temp_w, temp_b) and (beta_w, beta_b) vs. time for a case with non-zero equilibrium concentrations of both black and white daisies

Adaptive Stepsize in Runge-Kutta

Why Adaptive Stepsize?