Homework 4

ATSC 507

Christopher Rodell

$$T\,(^{\circ}C) = A\left(c\ m\ \Delta t + \ Tref\ - Tref\_o\right)\left(\ Tref\ \_o - c\ m\ \Delta t\right)$$

Given the above function with Tref_o = 2, A = 1, c = 1.5, Δt = 1, and t = m*Δt. This was the basis for the worksheet that I handed out today in class (the coloured curved lines in the fig below), where I used variable Tref in the range of 2 to 6, and m in the range of 0 to 1. Given the info above, the function that you should apply to the finite difference methods (a) - (d) below is:*

$$\partial T/\partial t = f(t, T) = 1.5\ \{2 - 1.5\ t - [T/(2 - 1.5\ t)]\}$$

Hint: To get the eq above for f(t, T), I first solved the eq above for Tref(T, t). Then I analytically found ∂T/∂t from the first equation above, and substituted in the expression for Tref. This takes advantage of the fact that Tref is constant along any of the curves in the fig below.*

Please start from initial condition of T = 2 degC as we did in class, but compute the new T (degC) at 1 timestep (1Δt) ahead using:

- a) Euler forward

$$T_{n+1} = f(T_n,\ t_n) * \Delta t + T_n$$

- b) Runge–Kutta 2nd order (mid-point)

$$T^* = T_n + \frac{\Delta t}{2}\ f\,(T_n, t_n, \cdots)$$
$$T_{n+1} = T_n + \Delta t\ f\left(T^*, t_n + \frac{\Delta t}{2}\ , \ldots .\right)$$

- c) Runge–Kutta 3rd order

$$T^* = T_n + \tfrac{\Delta t}{3}\ f(T_n\ , t_n\ , x_n)$$
$$T^{**} = T_n + \tfrac{\Delta t}{2}\ f\left(T^*, t_n + \tfrac{\Delta t}{3}\right)$$
$$T_{n+1} = T_n + \Delta t + \left(T^{**}, \quad t_n + \tfrac{\Delta t}{2}, \cdots\right)$$

- d) Runge–Kutta 4th order

$$k_1 = f(T_n\ ,\ x_n\ ,\ t_n)$$
$$k_2 = f\left(T_n + \tfrac{1}{2}\Delta t\ k_1,\ x_n\ ,\ t_n + \tfrac{1}{2}\Delta t\right)$$
$$k_3 = f\left(T_n + \tfrac{1}{2}\Delta t\ k_2,\ x_n\ ,\ t_n + \tfrac{1}{2}\Delta t\right)$$
$$k_4 = f\,(T_n + \Delta t\ k_3\ ,\ x_n\ ,\ t_n + \Delta t)$$
$$T_{n+1} = T_n + \tfrac{\Delta t}{6}\ \ (k_1 + 2k_2 + 2k_3 + k_4)$$

- e) Which one gave an answer closest to the actual analytical answer as given by the function above? (Note: do NOT use the 1-D model from the previous HW for this.)

  Runge–Kutta 3rd order gave the best approximation and was closest to the analytical solution.

In [1]:

```
import context
import numpy as np
from collections import namedtuple


class Approximator:

    def __init__(self, valueDict):
        self.__dict__.update(valueDict)

    def solution(self):
        T = self.A * ((self.c * self.m * self.dt) \
            + self.Tref - self.Tn) * (self.Tn - (self.c * self.m * self.dt))
        return T


    # function slope
    def TempFun(self):
        f = 1.5 * ( 2 - 1.5 * self.t - (self.Tn / (2 - (1.5 * self.t))))
        return f

    ## Euler forward
    def eulerf(self):
        T = self.TempFun() * self.dt + self.Tn
        return T

    ## Runge-Kutta 2nd order (mid-point)
    def rk2(self):
        Tn = self.Tn
        t = self.t

        T_str = Tn + (self.dt/2) * self.TempFun()

        self.Tn = T_str
        self.t  = (t + (self.dt/2))
        T  = Tn + self.dt * self.TempFun()

        self.Tn = Tn
        self.t  = t
        return round(T, 4)

    ## Runge-Kutta 3rd order
    def rk3(self):
        Tn = self.Tn
        t = self.t

        T_str = self.Tn + (self.dt/3) * self.TempFun()

        self.Tn = T_str
        self.t  = (t + (self.dt/3))
        T_str_str  = Tn + (self.dt/2) * self.TempFun()

        self.Tn = T_str_str
        self.t  = (t + (self.dt/2))
        T  = Tn + self.dt * self.TempFun()

        self.Tn = Tn
        self.t  = t
        return round(T, 4)
```

```
    ## Runge–Kutta 4th order
    def rk4(self):
        Tn = self.Tn
        t = self.t

        k1 = self.TempFun()

        self.Tn = (Tn + (self.dt/2)*k1)
        self.t  = (t + (self.dt/2))
        k2 = self.TempFun()

        self.Tn = (Tn + (self.dt/2)*k2)
        self.t  = (t + (self.dt/2))
        k3 = self.TempFun()

        self.Tn = (Tn + self.dt*k3)
        self.t  = (t + (self.dt))
        k4 = self.TempFun()

        T = Tn + (self.dt/6) * (k1 + (2 * k2) + (2 * k3) + k4)

        self.Tn = Tn
        self.t  = t
        return round(T, 4)
```

```
******************************
context imported. Front of path:
/Users/crodell/atsc507
/private/var/folders/9s/0p5yd78j0yd94hjttzwb6gq00000gp/T/f359e1ba-11
a8-4bdb-a61a-782a94744618
******************************

through /Users/crodell/atsc507/py/hw4/context.py -- pha
```

In [2]:

```
initialVals={'Tn': 2. , 't':0. , 'm':1. ,'dt':1. , 'A': 1. , 'Tref': 3. , 'c':
1.5 }
intVals= Approximator(initialVals)

T_dict = {}

T_dict.update({"Analytic Solution": intVals.solution()})
T_dict.update({"Euler Forward": intVals.eulerf()})
T_dict.update({"Runge–Kutta 2nd order": intVals.rk2()})
T_dict.update({"Runge–Kutta 3rd order": intVals.rk3()})
T_dict.update({"Runge–Kutta 4th order": intVals.rk4()})


print(T_dict)
```

```
{'Analytic Solution': 1.25, 'Euler Forward': 3.5, 'Runge–Kutta 2nd o
rder': 0.575, 'Runge–Kutta 3rd order': 1.625, 'Runge–Kutta 4th orde
r': 0.845}
```