Loyola University

Real Time and Embedded Systems

PRACTICE 1 RESULTS

Authors:
Martyna Baran
Zuzanna Jarlaczynska

December 8, 2023

Problem 1: global variables

In our program we have introduced global variables that are crucial for tasks execution. Constant matrices A and B are necessary for task 1 and matrix K for task 2. Current tanks levels and flows are stored correspondingly in arrays height and flows. Additionally, we also save flows rates and heights obtained in previous iteration. The mentioned arrays and height2 array are initialised with values adequate for the point of equilibrium.

global variables

```
#define SAMP_TIME 5
#define MAX_HEIGHT 1.3

RT_TASK task1, task2, task3;
RTIME sec = 1e9;
FILE *result_file;
RT_SEM sem1,sem2;

double A[4][4] = {{0.9419, 0.0,0.0401, 0.0},
                  {0.0, 0.9334, 0, 0.0380},
                  {0.0, 0.0, 0.9587, 0.0},
                  {0.0,0.0,0.0,0.9607}

};

double B[4][2] = {{0.0083, 0.0003},
                  {0.0004, 0.0111},
                  {0.0, 0.0168},
                  {0.0196, 0.0}
};


double K[2][4] ={{0,0.0000,0.0000,-7.5016},
                 {0,0.0000,-7.4553,0.0000}
};

double height[4] ={0.627,0.636,0.652,0.633};
double flows[2]={-7.7,-7.875};
double flows_prev[2] = {1.6429,2.0};
double height_prev[4] = {0};
double height2[4]= {0.65,0.66,0.65,0.66};
```

Problem 2: tanks levels

The first task is in charge of calculating and changing the tanks levels. The change is introduced by two given matrices: A and B. Firstly, we multiply this two matrices by adequately previous tanks levels and water flows. After the multiplication, the obtained values are summed and stored in additional array. Finally, the previous and current values are assigned to proper global arrays.

code for task 1

```
void fun1(void* arg){
    rt_task_sleep(sec);
    while(1){
        rt_sem_p(&sem1, TM_INFINITE);
        double h_new[4]={0.0,0.0,0.0,0.0};

        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++){
                h_new[i]+=A[i][j]*height[j];
            }
            for(int j = 0; j < 2; j++){
```

```
                h_new[i]+=B[i][j]*flows[j];
            }
            if(h_new[i] > MAX_HEIGHT){
                perror("level to high");
                return;
            }

        }

        for(int i = 0; i < 4; i++){
            height_prev[i]=height[i];
            height[i] = h_new[i];
        }
        rt_sem_v(&sem2);
        rt_task_wait_period(NULL);
    }
}
```

Problem 3: flow control

The seconds task aims to control the water flows. The calculations are made using the given matrix K and previous heights. After the multiplication, the obtained results are stored as new flows. Finally, we assign previous flows and obtained flows to proper global arrays.

code for task 2

```
void fun2(void *arg){
    rt_task_sleep(sec);
    while(1){
        rt_sem_p(&sem2, TM_INFINITE);
        double flows_new[2]={0.0,0.0};

        for(int i = 0; i < 2; i++){
            for(int j =0;j<4;j++){
                flows_new[i]+= K[i][j]*height[j];
            }

            if(flows_new[i] > 2.5){
                perror("flow to high");
                return;
            }
        }

        for(int i = 0; i < 2;i++){
            flows_prev[i]=flows[i];
            flows[i]= flows_new[i];
        }
        rt_task_wait_period(NULL);
    }
}
```

Problem 4: values printing

The role of the third task is to print the current tanks levels and flows. Height2 is the initial height so we need to add it to obtained heights. Finally, we save the printed values in result file.

C Code for task 3

```c
void fun3(void *arg){
    rt_task_sleep(sec);
    while(1){

        double H[4] ={0.0,0.0,0.0,0.0};
        for(int i=0;i<4;i++){
            H[i]= height[i]+height2[i];

        }
        printf("Tank Levels: h1=%f, h2=%f, h3=%f, h4=%f\n", H[0], H[1], H[2], H[3]);
        printf("Control Inputs: Qa=%f, Qb=%f\n", flows[0], flows[1]);
        fprintf(result_file, "Tank Levels: h1=%f, h2=%f, h3=%f, h4=%f\n", H[0],
         ↪ H[1], H[2], H[3]);
        fprintf(result_file, "Control Inputs: Qa=%f, Qb=%f\n", flows[0], flows[1]);
        fflush(result_file);
        rt_sem_v(&sem1);
        rt_task_wait_period(NULL);
    }
}
```

Problem 5: main function

To ensure the reliability of our program we must take into account that all 3 tasks need to access the same global variables - a shared resource. So as to avoid more than one task accessing the same variable simultaneously, we decided to introduce 2 semaphores. The sem1 semaphore is released in the end of printing task (task3). After that, task1 can change the tanks levels and then release the sem2. Finally, task 2 modifies water flows. All tasks are periodic with period 5s, so this cycle repeats until we meet certain flows or tanks levels.

C code for main

```c
void main(int agr, char* argv[]){
    result_file = fopen("results.txt", "a");
    if (result_file == NULL) {
        perror("Error opening file");
    }
    rt_task_create(&task1, "task1", 0,50,0);
    rt_task_create(&task2, "task2", 0,50,0);
    rt_task_create(&task3, "task3", 0,50,0);

    rt_task_set_periodic(&task1, TM_NOW, 5000000000);
    rt_task_set_periodic(&task2, TM_NOW, 5000000000);
    rt_task_set_periodic(&task3, TM_NOW, 5000000000);
    rt_sem_create(&sem1, "semaphore1", 0, S_FIFO);
    rt_sem_create(&sem2, "semaphore2", 0, S_FIFO);
    rt_task_start(&task1, &fun1,0);
    rt_task_start(&task2, &fun2, 0);
    rt_task_start(&task3, &fun3, 0);

    printf("End program by ctrl+C\n");
    pause();
    fclose(result_file);
    return;
}
```

Problem 6: results comparison

(1) Simulink model results

Tank Levels:	$h1 = 1.277,$	$h2 = 1.296,$	$h3 = 1.302,$	$h4 = 1.293$
Control Inputs:	$Qa = -4.749,$	$Qb = -4.861$		
Tank Levels:	$h1 = 1.2,$	$h2 = 1.188,$	$h3 = 1.143,$	$h4 = 1.117$
Control Inputs:	$Qa = -3.43,$	$Qb = -3.673$		
Tank Levels:	$h1 = 1.14,$	$h2 = 1.102,$	$h3 = 1.022,$	$h4 = 0.9904$
Control Inputs:	$Qa = -2.478,$	$Qb = -2.776$		
Tank Levels:	$h1 = 1.091,$	$h2 = 1.034,$	$h3 = 0.9314,$	$h4 = 0.8987$
Control Inputs:	$Qa = -1.79,$	$Qb = -2.098$		
Tank Levels:	$h1 = 1.051,$	$h2 = 0.9799,$	$h3 = 0.8626,$	$h4 = 0.8324$
Control Inputs:	$Qa = -1.293,$	$Qb = -1.585$		
Tank Levels:	$h1 = 1.018,$	$h2 = 0.9361,$	$h3 = 0.8107,$	$h4 = 0.7846$
Control Inputs:	$Qa = -0.9344,$	$Qb = -1.198$		
Tank Levels:	$h1 = 0.9901,$	$h2 = 0.9005,$	$h3 = 0.7714,$	$h4 = 0.75$
Control Inputs:	$Qa = -0.6751,$	$Qb = -0.9054$		
Tank Levels:	$h1 = 0.9655,$	$h2 = 0.8713,$	$h3 = 0.7418,$	$h4 = 0.0725$
Control Inputs:	$Qa = -0.4877,$	$Qb = -0.6842$		
Tank Levels:	$h1 = 0.9439,$	$h2 = 0.8472,$	$h3 = 0.7194,$	$h4 = 0.707$
Control Inputs:	$Qa = -0.3523,$	$Qb = -0.5171$		
...				

(2) Obtained results

Tank Levels:	$h1 = 1.277,$	$h2 = 1.296,$	$h3 = 1.302,$	$h4 = 1.293$
Control Inputs:	$Qa = -4.748513,$	$Qb = -4.860856$		
Tank Levels:	$h1 = 1.195444,$	$h2 = 1.617203,$	$h3 = 1.142772,$	$h4 = 1.117203$
Control Inputs:	$Qa = -3.429755,$	$Qb = -3.673766$		
Tank Levels:	$h1 = 1.158654,$	$h2 = 1.307227,$	$h3 = 1.060702,$	$h4 = 1.032012$
Control Inputs:	$Qa = -2.790684,$	$Qb = -3.061904$		
Tank Levels:	$h1 = 1.121489,$	$h2 = 1.387122,$	$h3 = 1.010702,$	$h4 = 0.962694$
Control Inputs:	$Qa = -2.270692,$	$Qb = -2.551947$		
Tank Levels:	$h1 = 1.08821,$	$h2 = 1.406805,$	$h3 = 0.93529,$	$h4 = 0.906401$
Control Inputs:	$Qa = -1.847591,$	$Qb = -2.126922$		
Tank Levels:	$h1 = 1.058217,$	$h2 = 1.396805,$	$h3 = 0.887775,$	$h4 = 0.86006$
Control Inputs:	$Qa = -1.503327,$	$Qb = -1.772686$		
Tank Levels:	$h1 = 1.031025,$	$h2 = 1.355102,$	$h3 = 0.848174,$	$h4 = 0.82606$
Control Inputs:	$Qa = -1.22321,$	$Qb = -1.477446$		
Tank Levels:	$h1 = 1.006238,$	$h2 = 1.356155,$	$h3 = 0.815168,$	$h4 = 0.792677$
Control Inputs:	$Qa = -0.995287,$	$Qb = -1.231379$		
Tank Levels:	$h1 = 0.983534,$	$h2 = 1.53671,$	$h3 = 0.78766,$	$h4 = 0.757955$
Control Inputs:	$Qa = -0.809834,$	$Qb = -1.026294$		
...				

Problem 7: conclusions

As we can see, the obtained results are very similar to the model outputs, but not the same. At the beginning both tanks levels and control flows are identical. However, our simulation is less precise than the program. It tends to round the results to maximum 4 decimal places. In contrast, our program allows up to 6 decimal places. Although it doesn't matter much at first, after just a few iterations the results differ from each other. Initially small differences increase with each task completion. Therefore, the more iterations we go through, the bigger becomes the error. After all, we consider our results as acceptable and correct. What could be improved is the precision of simulation and initial values.