Exam in Practical Programming

Amalie Louise Stokholm, 201303663, stokholm@phys.au.dk
March 22, 2017

Will these functions leak memory?

```c
void stat() {
	double a[500];
}
```

The functions as `stat` stores the temporary variables in the stack. The stack has size limits, but a size of 500 is probably not big enough to hit this limit. The stack grows and shrinks in memory as the function pushes more variables to it, but the stack is user-friendly as the variables are allocated and freed automatically. As soon as the function has finished, the variables are freed from the stack, the memory is re-usable, and thus there is no memory leak.

```c
void vla(size_t n) {
	double a[n];
}
```

This function `vla` uses as `stat` the stack to store memory. When the function is done, the variables are freed from the stack and the memory can be reused. Just as `stat`, this function does not leak memory.

```c
void mal() {
	double* a = (double*) malloc(500 * sizeof(double));
}
```

In contrast to `stat` and `vla`, the next couple of functions use the heap, where memory is not managed automatically for the user. The user needs to manually allocate and free memory in order to avoid memory leak. In `mal`, the memory is only allocated, but not freed, and therefore there is a memory leak as this leads to a build-up of non-reusable memory.

```c
void maf() {
	double* a = (double*) malloc(500 * sizeof(double));
	free(a);
}
```

This function `maf` uses the heap, but in contrast to `mal`, the memory is both allocated and freed. A memory leak is thus avoided as all the memory can be reused.

```c
double* mar() {
	double* a = (double*) malloc(500 * sizeof(double));
	return a;
}
```

In `mar`, there is no memory leak as it returns a pointer. A function calling this function can reuse the memory allocated by a by using this pointer.

Problem

Implement the Bessel function of the first kind of integer index using Bessel's integral representation.

$$J_n(x) = \frac{1}{\pi}\int_0^\pi \cos(nt - x\sin(t))\,\mathrm{d}t. \tag{1}$$

Compare with the corresponding function from `<math.h>` or from GSL.

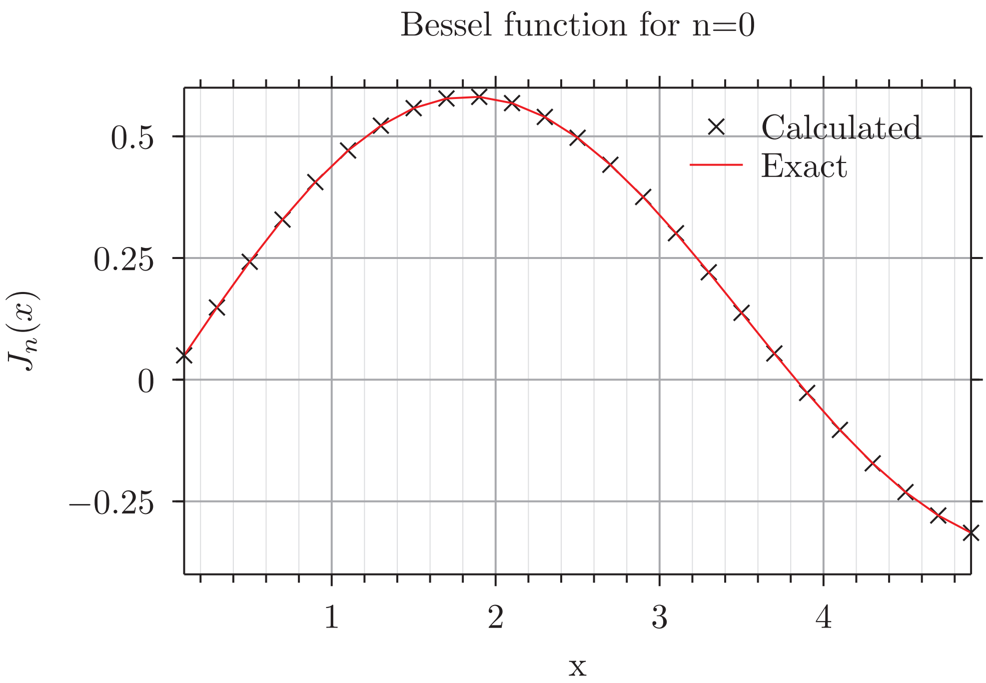

Figure 1: Comparison between the calculated $J_n(x)$ (points) and $J_n(x)$ from `<math.h>` (lines) for a given n.