Numerical Algorithms for Visual Computing III 2011
Example Solutions for Assignment 3

Problem 1 (Slow starters)

1. The function $f(x, y, z) = 2x^2 + 3y^2 + z^2$ has the gradient $\nabla f(x, y, z) = (4x, 6y, 2z)^\top$. Evaluated at point $(x, y, z)^\top = (2, 1, 3)^\top$, we have $\nabla f((2, 1, 3)^\top) = (8, 6, -6)^\top$. Computing the directional derivative$\nabla_v f$ in direction $v = (1, 0, -2)^\top$ gives the following result:

$$(1, 0, -2) \cdot (8, 6, -6)^\top = -4.$$

At this point, we should note, that it is mathematically more sound to only consider v as a normalised vector, i.e. $v^\top \nabla f = \frac{-4}{\sqrt{5}}$.

2. The function $f(x, y) = \exp(x)\cos(x)$ has the gradient $\nabla f(x, y) = (\exp(x)\cos(y), -\exp(x)\sin(y))^\top$. Evaluated at point $(x, y)^\top = (2, \pi)^\top$, we have $\nabla f((2, \pi)^\top) = (-\exp(2), 0)^\top$. Computing the directional derivative$\nabla_v f$ in the normalised direction $v = (2, 3)^\top$ gives the following result:

$$\frac{1}{\sqrt{13}}(2, 3) \cdot (-\exp(2), 0)^\top = \frac{-2\exp(2)}{\sqrt{13}}.$$

Problem 2 (Germany's Next Top Stencil)

a) The given energy functional has the Euler-Lagrange equation

$$\frac{\mathrm{d}}{\mathrm{d}x} u'(x) = u''(x) \stackrel{!}{=} 0.$$

b) A function that satisfies this constraints is given by a simple linear equation $ax + b$. Given the boundary conditions $u(1) = 10$ and $u(11) = 20$, $f(x) = x + 9$ satisfies these constraints.

c) At first, as a reference, let us discretise the Euler-Lagrange equation with mixing forward-backward discretisation for the different derivatives:

$$\left.\frac{\mathrm{d}}{\mathrm{d}x} u'\right|_{x=j} \approx \frac{[u']|_{x=j+1} - [u']|_{x=j}}{\Delta x} \approx \frac{\left(\frac{u_{j+1} - u_j}{\Delta x}\right) - \left(\frac{u_j - u_{j-1}}{\Delta x}\right)}{\Delta x} = \frac{u_{j+1} - 2u_j + u_{j-1}}{(\Delta x)^2}$$

This is the standard way to compute an approximation for the second derivative with a second order error term. Implementing this method as an iterative scheme, i.e.

$$u_j^{k+1} \;=\; u_j^k + \tau \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{(\Delta x)^2}$$

gives the desired result as proposed in part (b). This was just for a reference. Let us now go ahead with the implementation with only forward derivatives:

$$\frac{\mathrm{d}}{\mathrm{d}x} u' \Big|_{x=j} \;\approx\; \frac{[u']|_{x=j+1} - [u']|_{x=j}}{\Delta x} \approx \frac{\left(\frac{u_{j+2}-u_{j+1}}{\Delta x}\right) - \left(\frac{u_{j+1}-u_j}{\Delta x}\right)}{\Delta x} = \frac{u_{j+2} - 2u_{j+1} + u_j}{(\Delta x)^2}$$

Implemented as an iterative scheme this algorithm results in

$$u_j^{k+1} \;=\; u_j^k + \tau \frac{u_{j+2}^k - 2u_{j+1}^k + u_j^k}{(\Delta x)^2}.$$

The problem with this scheme is given by the fact that the left boundary condition is not used at all and an additional boundary condition $u(12)$ is needed. If one chooses for example $u(12) = 20$, then the signal is becoming $(10, 20, 20, \ldots, 20)$, i.e. it interpolates a straight line between points $u(11)$ and $u(12)$. Similarly this phenomenon appears with a derivation with only backward derivatives, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}x} u' \Big|_{x=j} \;\approx\; \frac{[u']|_{x=j} - [u']|_{x=j-1}}{\Delta x} \approx \frac{\left(\frac{u_j-u_{j-1}}{\Delta x}\right) - \left(\frac{u_{j-1}-u_{j-2}}{\Delta x}\right)}{\Delta x} = \frac{u_j - 2u_{j-1} + u_{j-2}}{(\Delta x)^2}$$

resulting in the iterative scheme

$$u_j^{k+1} \;=\; u_j^k + \tau \frac{u_j^k - 2u_{j-1}^k + u_{j-2}^k}{(\Delta x)^2}.$$

With this implementation we have to introduce an additional boundary condition at $u(0)$. Coming to the last candidate, i.e. the central difference scheme, we have an additional problem: With the derivation

$$\frac{\mathrm{d}}{\mathrm{d}x} u' \Big|_{x=j} \;\approx\; \frac{[u']|_{x=j+1} - [u']|_{x=j-1}}{2\Delta x} \approx \frac{\left(\frac{u_{j+2}-u_j}{2\Delta x}\right) - \left(\frac{u_j-u_{j-2}}{2\Delta x}\right)}{\Delta x} = \frac{u_{j+2} - 2u_j + u_{j-2}}{(4\Delta x)^2}$$

and the resulting iterative scheme

$$u_j^{k+1} \;=\; u_j^k + \tau \frac{u_{j+2}^k - 2u_j^k + u_{j-2}^k}{(4\Delta x)^2},$$

we see that the signal is split in two signals actually that are not communicating with each other. Also we need here four boundary conditions defined at $u(0), u(1), u(11), u(12)$. With a bad choice of boundary conditions it is possible to have two straight lines that are interchanging between even and odd signal parts.

Problem 3 (Tortellini and Ambrosia Revisited) For the 1-D AT energy functional we use the discretisation scheme from the last assignment sheet and implement this with Neumann boundary conditions. In order to have no problems with the iterative scheme, one should use a small time step size. u is being initialised with the original signal and v with a simple signal with only 1s.

Problem 4 (Rationalising with Gold) This method is also known as the "golden section method". It is a variant of a bisection method. Let $f : [a, b] \to \mathbb{R}$ be a unimodal function with a minimum ξ and $a < x_1 < x_2 < b$. As we assume f to be monotonous we can imply

$$f(x_1) \geq f(x_2) \quad \Rightarrow \quad \xi \in [x_1, b] \tag{1}$$
$$f(x_1) < f(x_2) \quad \Rightarrow \quad \xi \in [a, x_2] \tag{2}$$

The idea is that in such a case, one can use a smaller interval $[a^1, b^1] \subset [a, b]$ with $[a^0, b^0] := [a, b]$ and we can set

$$[a^1, b^1] \quad := \quad \begin{cases} [x_1, b] & \text{if } f(x_1) \geq f(x_2) \\ [a, x_2] & \text{if } f(x_1) < f(x_2) \end{cases} \tag{3}$$

Together with the conditions described in the exercise parts (a) and (b), we can compute an optimal value for τ. Assume that we have a minimum in the interval $[x_1, b]$. By the condition $b^i - a^i = \tau(b^{i-1} - a^{i-1})$ we know that the interval $[x_1, b]$ has the size $\tau(b-a)$ and therefore $[a, x_1]$ has size $(1-\tau)(b-a)$. From the settings, one can check that in the next step, the old x_2^{i-1} becomes the new x_1^i, which means that the interval $[x_1^{i-1}, x_2^{i-1}]$ is of size $\tau(1-\tau)(b-1)$ and $[x_2^{i-1}, b^{i-1}] = \tau^2(b-a)$. As we know that the length of $[a, x_1]$ is equal to the length of $[x_2, b]$, we are dealing with an equation $1 - \tau = \tau^2$. This is a quadratic equation and has the solutions $\frac{\pm\sqrt{5}-1}{2}$. As one of the solutions is negative (and therefore useless), the optimal solution is $\tau = \frac{\sqrt{5}-1}{2}$, which is also known as "golden section".

From this knowledge we can devise an algorithm, which is known as the so-called "golden section method". Initially we have the following settings:

$$\begin{aligned}
\tau &= \frac{\sqrt{5}-1}{2} \\
x_1 &= b^0 - \tau(b^0 - a^0) \\
x_2 &= a^0 - \tau(b^0 - a^0) \\
f_a^0 &= f(a^0) \\
f_b^0 &= f(b^0) \\
f_1^0 &= f(x_1^0) \\
f_2^0 &= f(x_2^0)
\end{aligned}$$

Set $k = 0$.

`while` $b^k - a^k > \varepsilon$
`if` $f_1^k \geq f_2^k$
$a^{k+1} = x_1^k$
$b^{k+1} = b^k$
$x_1^{k+1} = x_2^k$
$x_2^{k+1} = a^{k+1} + \tau(b^{k+1} - a^{k+1})$
$f_a^{k+1} = f_1^k$
$f_b^{k+1} = f_b^k$
$f_1^{k+1} = f_2^k$
$f_2^{k+1} = f(x_2^{k+1})$
`else`
$a^{k+1} = a^k$
$b^{k+1} = x_2^k$
$x_2^{k+1} = x_1^k$
$x_1^{k+1} = b^{k+1} - \tau(b^{k+1} - a^{k+1})$
$f_a^{k+1} = f_a^k$
$f_b^{k+1} = f_2^k$
$f_1^{k+1} = f_1^k$
$f_2^{k+1} = f(x_1^{k+1})$
`end`
$k = k + 1$
`end`

For example, by choosing as a starting interval $[-4, 0]$, the golden section method needs 32 iterations for the result -2.5687298.