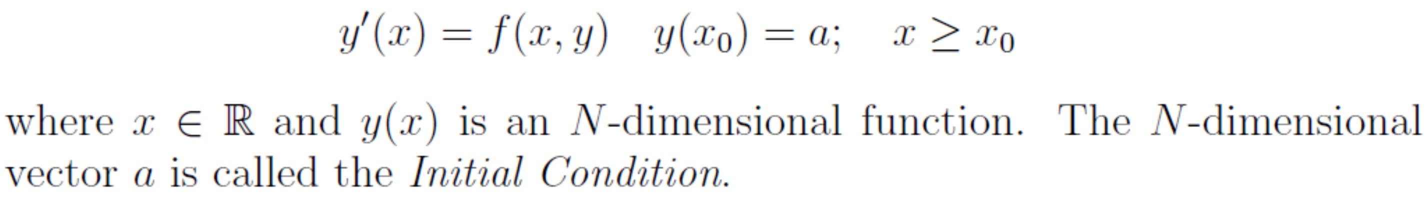
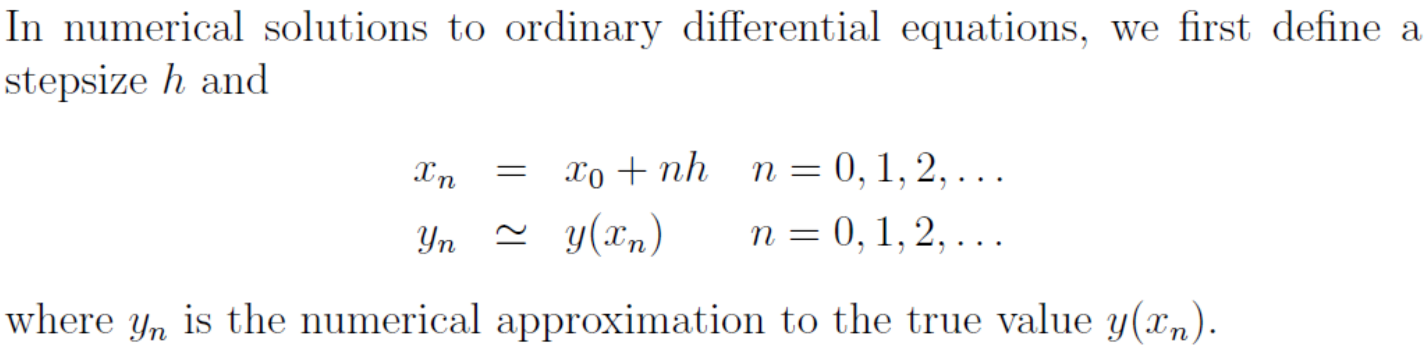
Initial value problem using vector notation:



$$y'(x) = f(x,y) \quad y(x_0) = a; \quad x \geq x_0$$

where $x \in \mathbb{R}$ and $y(x)$ is an N-dimensional function. The N-dimensional vector a is called the *Initial Condition.*

Numerical solution:



In numerical solutions to ordinary differential equations, we first define a stepsize h and

$$\begin{aligned} x_n &= x_0 + nh \quad n = 0, 1, 2, \ldots \\ y_n &\simeq y(x_n) \qquad n = 0, 1, 2, \ldots \end{aligned}$$

where y_n is the numerical approximation to the true value $y(x_n)$.

1st order Runge-Kutta (Euler):

$$y_{n+1} = y_n + hf(x_n, y_n) + O(h^2)$$

2nd order Runge-Kutta (Midpoint):

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned}$$

4th order Runge-Kutta:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\ k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4 + O(h^5) \end{aligned}$$

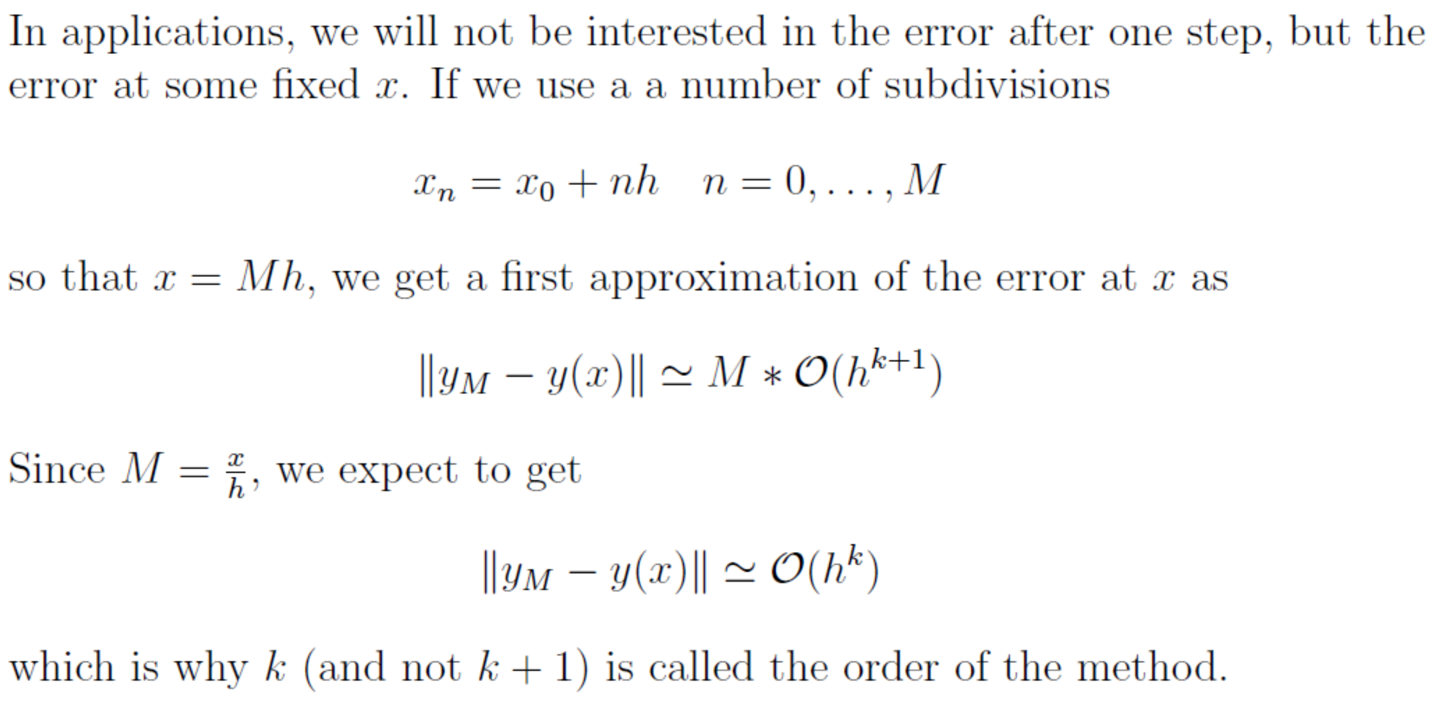
Trapezoidal method (implicit):

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right) + \mathcal{O}(h^3)$$

Leap-frog:

$$\xcancel{y_{n+1} = y_{n-1} + 2hf(x_n, y_n) + \mathcal{O}(h^3)}$$

Order of a numerical method (global order):



In applications, we will not be interested in the error after one step, but the error at some fixed x. If we use a a number of subdivisions

$$x_n = x_0 + nh \quad n = 0, \ldots, M$$

so that $x = Mh$, we get a first approximation of the error at x as

$$\|y_M - y(x)\| \simeq M * \mathcal{O}(h^{k+1})$$

Since $M = \frac{x}{h}$, we expect to get

$$\|y_M - y(x)\| \simeq \mathcal{O}(h^k)$$

which is why k (and not $k+1$) is called the order of the method.

The term for one step $\mathcal{O}(h^{k+1})$ is called the ”local order” of the method

Properties of differential equations with respect to perturbations

Consider now a small perturbation δ_0 on the initial condition

$$y(x_0) = a + \delta_0$$

We then get a new solution $y(x) + \delta(x)$ where $\delta(x)$ is the resulting perturbation at x. We will now have a differential equation for $y + \delta$

$$(y'(x) + \delta'(x)) = f(y(x) + \delta(x)) \quad x \geq x_0$$

We may now Taylor expand the right hand side around $y(x)$:

$$\cancel{y'(x)} + \delta'(x) \simeq \cancel{f(y(x))} + J(y(x))\delta(x)$$

where $J(y)$ is the Jacobian to $f(y)$ (remember f contains N functions in the N dimensional vector y).

We then get a differential equation for $\delta(x)$

$$\delta'(x) \simeq J(y(x))\delta(x)$$

$$\delta'(x) \simeq J(y(x))\delta(x)$$

Assume now for analysis that $N = 1$. If now $J(y(x)) \geq k > 0$ for a while along the trajectory, say between $x = u$ and $x = u + X$, we get

$$|\delta(u + X)| \geq e^{kX}|\delta(u)|$$

Hence, the perturbation leads to exponential separation. If this occurs for a sufficient big interval size X, it will be difficult or impossible to obtain any global accuracy after this interval.

For $N > 1$, we remember that if for an $N \times N$ matrix A, we have that

$$Az = \lambda z$$

where $\lambda \in \mathbb{C}$, we say that λ is an *Eigenvalue* for A with *corresponding Eigenvector* z.

If we now for simplicity assume that $J(y(x)) \equiv A$ is constant, the same problem of exponential separation will occur if A has an Eigenvalue with positive real part and there is a component from the associated Eigenvector in the perturbation.

$$\begin{aligned} y_1'(x) &= y_1(x)y_2(x) \\ y_2'(x) &= -y_1'(x)^2 \end{aligned}$$

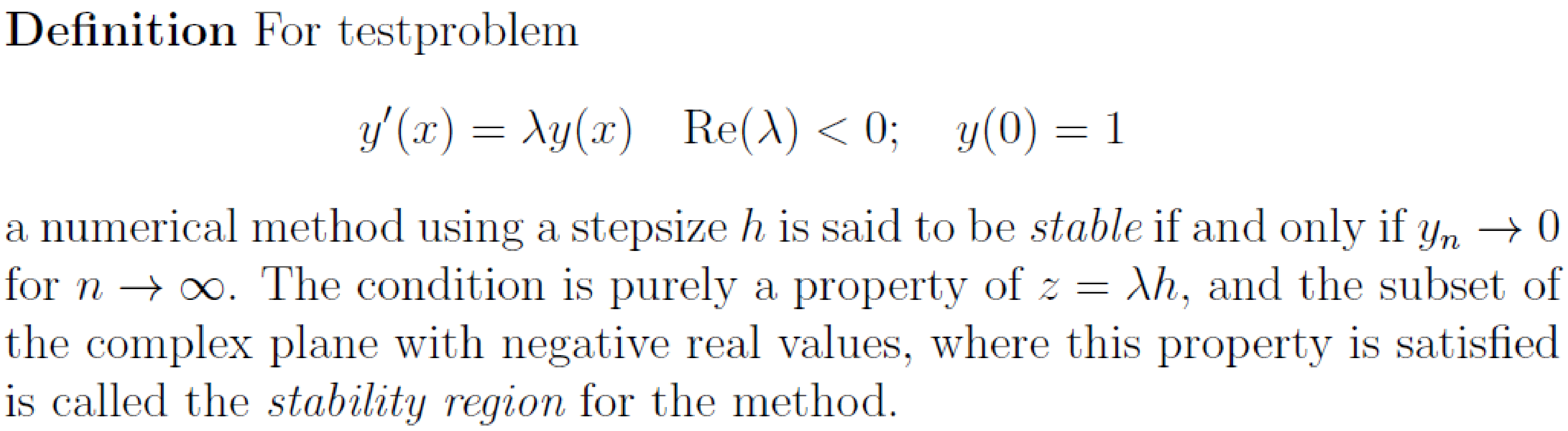
$$J(y(x)) = \begin{pmatrix} y_2(x) & y_1(x) \\ -2y_1(x) & 0 \end{pmatrix}$$

We then get Eigenvalues

$$\begin{aligned} \lambda_1 &= \frac{1}{2}\left(y_2(x) + \sqrt{y_2(x)^2 - 8y_1(x)^2}\right) \\ \lambda_2 &= \frac{1}{2}\left(y_2(x) - \sqrt{y_2(x)^2 - 8y_1(x)^2}\right) \end{aligned}$$

In the numerical solution, we quickly get $y_2(x) < 0$, in which case none of the Eigenvalues will have positive real values. The solutions therefore do not separate, and actually converge to the stable point $(y_1, y_2) = (0, -\sqrt{2})$ where the Eigenvalues are $\{0, -\sqrt{2}\}$.

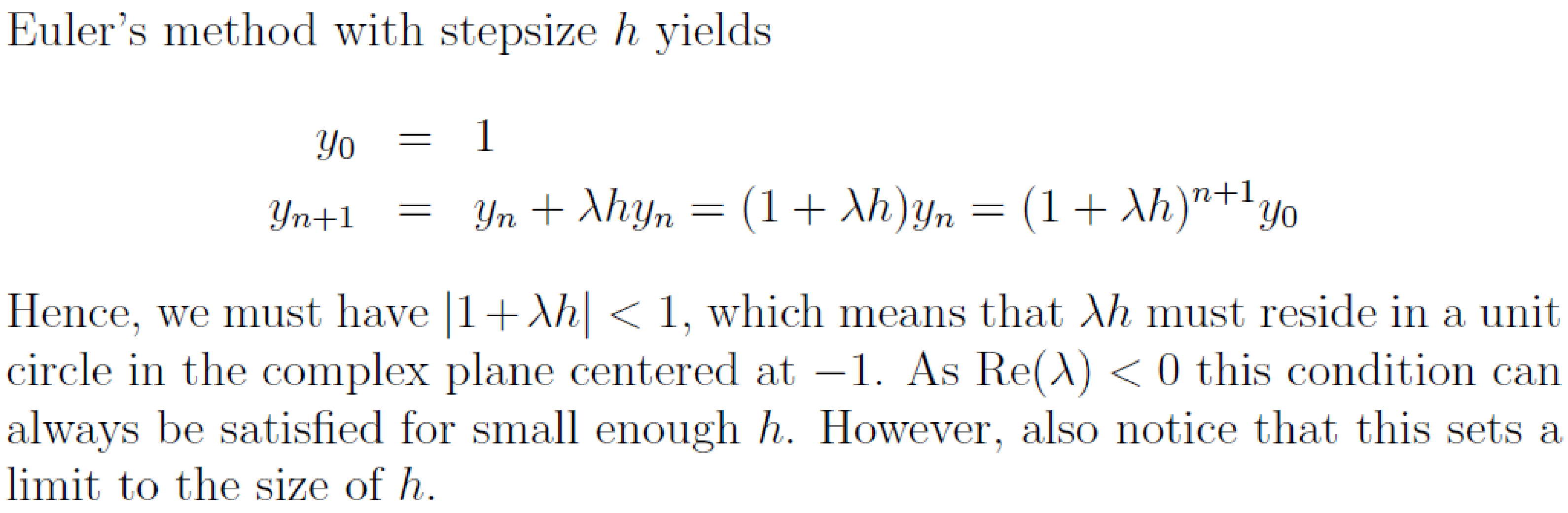
Numerical stability



Definition For testproblem

$$y'(x) = \lambda y(x) \quad \mathrm{Re}(\lambda) < 0; \quad y(0) = 1$$

a numerical method using a stepsize h is said to be *stable* if and only if $y_n \to 0$ for $n \to \infty$. The condition is purely a property of $z = \lambda h$, and the subset of the complex plane with negative real values, where this property is satisfied is called the *stability region* for the method.



Euler's method with stepsize h yields

$$\begin{aligned} y_0 &= 1 \\ y_{n+1} &= y_n + \lambda h y_n = (1 + \lambda h) y_n = (1 + \lambda h)^{n+1} y_0 \end{aligned}$$

Hence, we must have $|1 + \lambda h| < 1$, which means that λh must reside in a unit circle in the complex plane centered at -1. As $\mathrm{Re}(\lambda) < 0$ this condition can always be satisfied for small enough h. However, also notice that this sets a limit to the size of h.

Consider the second order Runge-Kutta method (midpoint method):

$$y_{n+1} = y_n + \lambda h y_{n+\frac{1}{2}} = y_n + \lambda h(y_n + \frac{1}{2}\lambda h y_n) = \left(\frac{1}{2}(\lambda h)^2 + (\lambda h) + 1\right) y_n$$

Hence, we must have

$$\left|\frac{1}{2}(\lambda h)^2 + (\lambda h) + 1)\right| < 1$$

Consider now the Trapezoidal method:

$$y_{n+1} = y_n + \lambda h \frac{1}{2}(y_{n+1} + y_n)$$

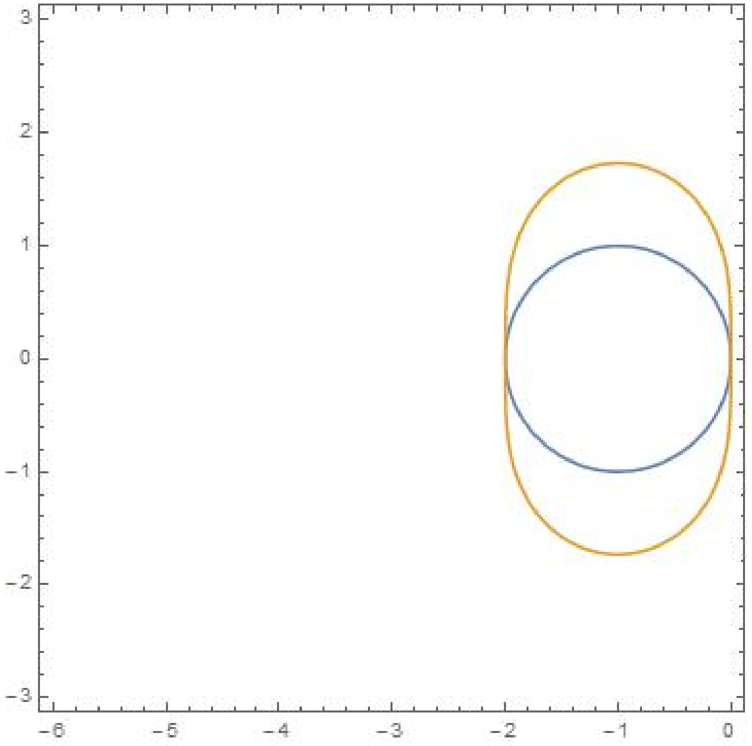
which can be reordered to

$$(1 - \frac{\lambda h}{2})y_{n+1} = (1 + \frac{\lambda h}{2})y_n$$

which yields

$$\left|\frac{1 + \frac{\lambda h}{2}}{1 - \frac{\lambda h}{2}}\right| < 1$$

Hence the stability region is the whole complex half plane with negative real values and hence the Trapezoidal method is numerical stable for all h-values. In particular if we have differential equations, where we know that $\text{Re}(\lambda) << 0$ (stiff differential equations), this method becomes relevant.

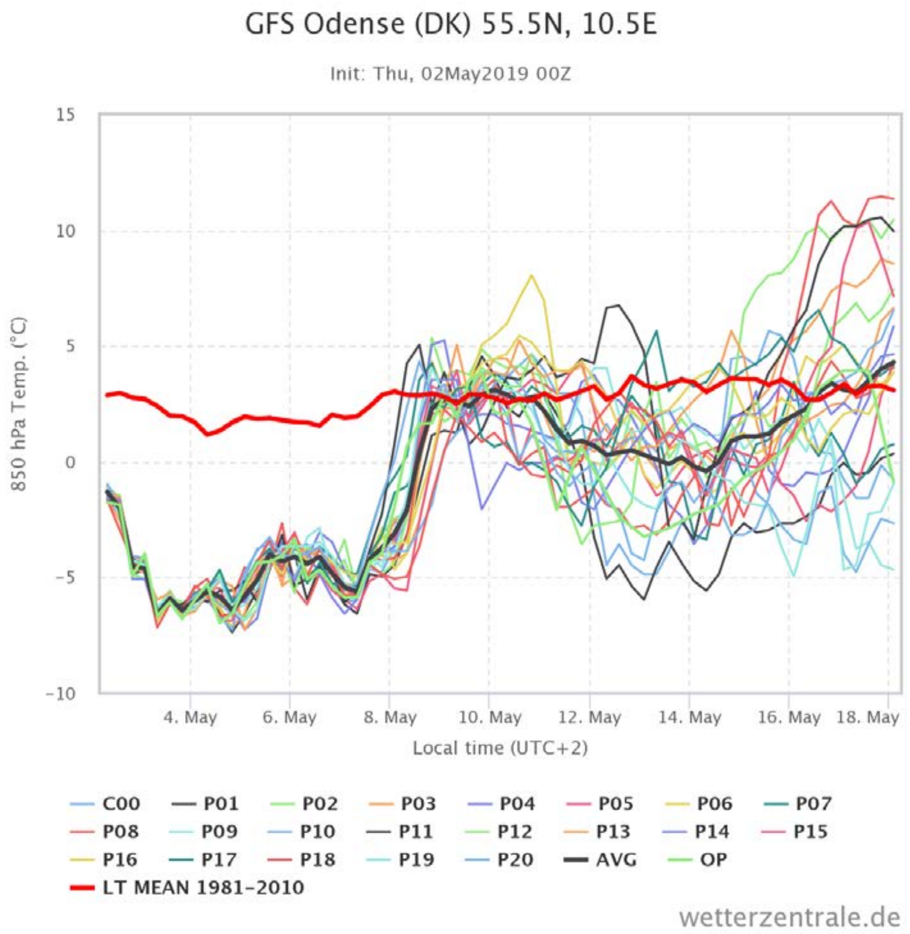


Stability region for Euler (blue) and Midpoint (orange)

Consider now the Leap-frog method:

$$y_{n+1} = 2\lambda h y_n + y_{n-1}$$

An analysis, which is a bit more involved than for the other methods above, yields that the stability region is EMPTY !!! Hence, don't use this method except in special cases where you are not interested in global accuracy, but only in e.g. obtaining statistical behavior over long runs with differential equations where perturbations lead to fast separation (e.g. weather forecasts).



A final remark:

In practise (for $N >> 1$), separation of trajectories and numerical stability cannot be studied via the Jacobian, so all the above is for theoretical study. Use Richardson Extrapolation to obtain global accuracy, and if a convergence cannot be observed, you will know that the global result cannot be trusted.

Two point boundary value problem. We consider the model problem:

$$\begin{cases} y''(x) = F(y'(x), y(x), x) \ \text{ for } \ a < x < b \\ y(a) = \alpha, \quad y(b) = \beta \end{cases}$$

Choose $y'(x) = s$. Then solve the initial value problem

$$\begin{aligned} y_1'(x) &= y_2(x) \quad y_1(a) = \alpha \\ y_2'(x) &= F(y_2(x), y_1(x), x) \quad y_2(a) = s \end{aligned}$$

with a some selected method and a stepsize h so that $b - a = Mh$ for some integer M. You then obtain a numerical approximation $(y_1)_M$ to $y_1(b)$. This is a function of the "shooting value" s and the stepsize h. We now define the deviation between the found numerical approximation and the desired value

$$\phi_h(s) = y_M - \beta$$

We now wish to solve

$$\phi_h(s) = 0$$

This leads to an algorithm called ”Shooting”. Select h and two shooting values s_0 and s_1, and compute $\phi_h(s_0)$ and $\phi_h(s_1)$ by integrating the ODE's.

Then compute

$$s_2 = s_1 - \phi_h(s_1)\frac{s_1 - s_0}{\phi_h(s_1) - \phi_h(s_0)}$$

which you will recognize as the Secant method. Now use s_2 as the new shooting value, integrate the ODE's to obtain $\phi_h(s_2)$ and do Secant with s_1 and s_2.

When the Secant method converges, you have found a solution that still depends on h. You may then divide the stepsize by 2, and run again. Here, you can use the last two s_i's from the old stepsize as the initial guesses for the Secant method.