Systems of non-linear equations

A typical problem gives N functional relations to be zeroed, involving variables $x_i, i = 0, 1, \ldots, N-1$:

$$F_i(x_0, x_1, \ldots, x_{N-1}) = 0 \qquad i = 0, 1, \ldots, N-1. \tag{9.6.2}$$

We let $\mathbf{x}$ denote the entire vector of values x_i and $\mathbf{F}$ denote the entire vector of functions F_i. In the neighborhood of $\mathbf{x}$, each of the functions F_i can be expanded in Taylor series:

$$F_i(\mathbf{x} + \delta\mathbf{x}) = F_i(\mathbf{x}) + \sum_{j=0}^{N-1} \frac{\partial F_i(\mathbf{x})}{\partial x_j} \delta x_j + \mathcal{O}(\|\delta\mathbf{x}\|^2) \tag{9.6.3}$$

The matrix of partial derivatives appearing in equation (9.6.3) is the *Jacobian* matrix $\mathbf{J}$:

$$J_{ij} \equiv \frac{\partial F_i}{\partial x_j}. \tag{9.6.4}$$

Jacobian

$$\mathbf{F}(\mathbf{x}+\delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x} + \mathcal{O}(\|\delta\mathbf{x}\|^2) \tag{9.6.5}$$

$$\mathbf{J}(\mathbf{x})\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}) \tag{9.6.6}$$

Matrix equation (9.6.6) can be solved by LU decomposition as described in §2.3. The corrections are then added to the solution vector,

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x} \tag{9.6.7}$$

Newtons method in one dimension:

$$x_{i+1} = x_i - \frac{1}{f'(x_i)} f(x_i)$$

Can be rewritten as:

$$\begin{aligned} f'(x_i)\Delta x &= -f(x_i) \\ x_{i+1} &= x_i + \Delta x \end{aligned}$$

Example of computing a Jacobian:

3 nonlinear equations in 3 unknowns

$$
\begin{aligned}
F_0(x_0, x_1, x_2) &= 2x_0^3 + x_1 \cos(x_2) &= 0 \\
F_1(x_0, x_1, x_2) &= x_0 x_1 + \sin(x_2) &= 0 \\
F_2(x_0, x_1, x_2) &= \sin(x_0) + x_1 x_2^2 &= 0
\end{aligned}
$$

The resulting Jacobian is

$$
\mathbf{J}(x_0, x_1, x_2) = \begin{pmatrix} 6x_0^2 & \cos(x_2) & -x_1 \sin(x_2) \\ x_1 & x_0 & \cos(x_2) \\ \cos(x_0) & x_2^2 & 2x_1 x_2 \end{pmatrix}
$$

Three non-linear equations in three unknowns

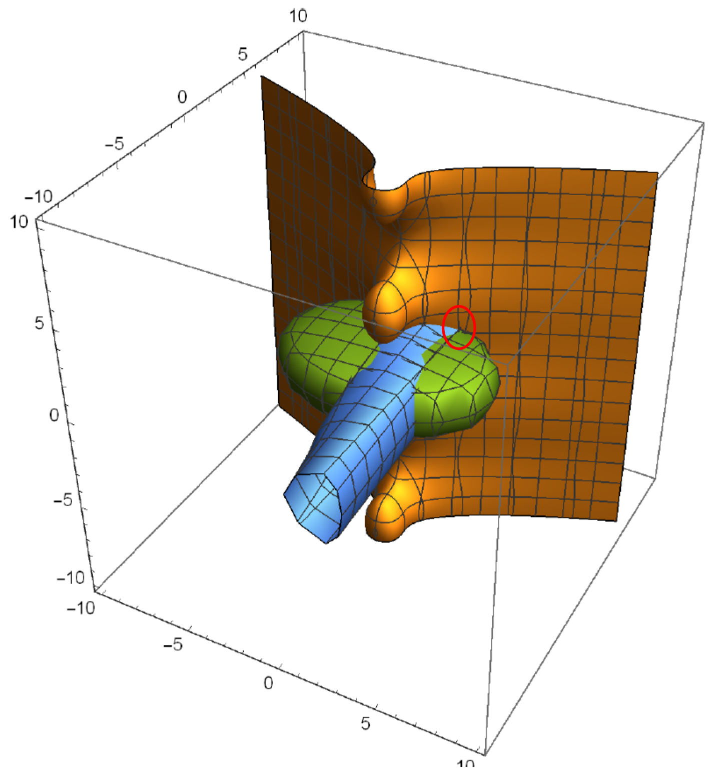
In[4]:= `f1[x1_, x2_, x3_] = 10 x1^2 - 5 x2^3 + 10 Cos[x3];` Orange

In[5]:= `f2[x1_, x2_, x3_] = (x1 - 1)^4 - 2 x2 + 4 x3^2 + x1 x2 - 15;` Blue

In[6]:= `f3[x1_, x2_, x3_] = x1^2 + 2 x2^2 + 3 x3^4 - 30;` Green

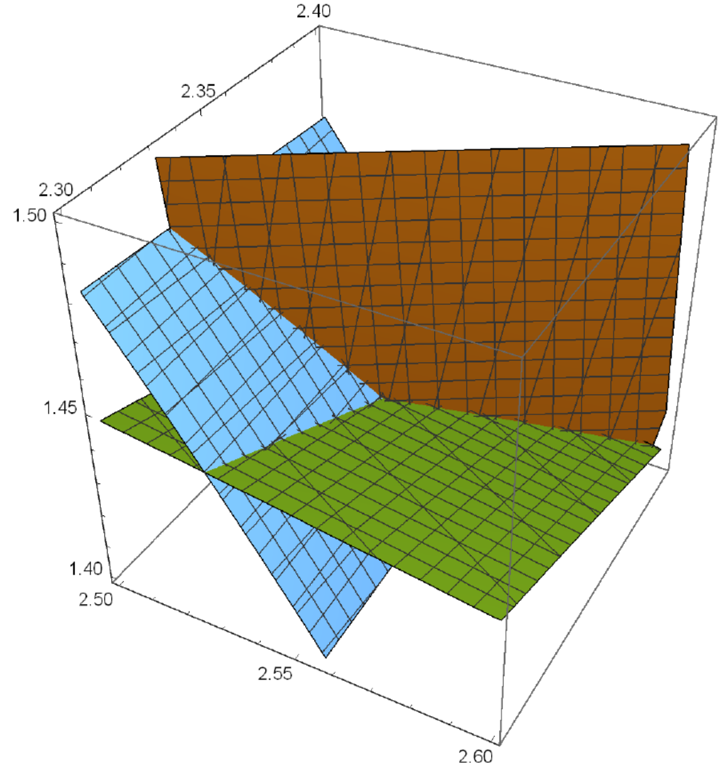
In[13]:= `ContourPlot3D[{f1[x1, x2, x3] == 0, f2[x1, x2, x3] == 0, f3[x1, x2, x3] == 0}, {x1, -10, 10}, {x2, -10, 10}, {x3, -10, 10}]`

Out[13]=



In[15]:= `ContourPlot3D[{f1[x1, x2, x3] == 0, f2[x1, x2, x3] == 0, f3[x1, x2, x3] == 0}, {x1, 2.5, 2.6}, {x2, 2.3, 2.4}, {x3, 1.4, 1.5}]`

Out[15]=



Overdetermined systems of non-linear equations. Example: Real robot calibration…

Jogging a calibration pointer mounted on the robot tool (in this case a nine axis robot) to a given position (top of cone) with different joint configurations

$$
\begin{aligned}
f(\mathbf{x}, \mathbf{q}_1) - \mathbf{p} &= \mathbf{0} \\
f(\mathbf{x}, \mathbf{q}_2) - \mathbf{p} &= \mathbf{0} \\
&\dots \\
&\dots \\
&\dots \\
f(\mathbf{x}, \mathbf{q}_K) - \mathbf{p} &= \mathbf{0}
\end{aligned}
$$

where $f(\mathbf{x}, \mathbf{q})$ is the position part of the forward kinematics for a joint configuration $\mathbf{q}$, and $\mathbf{x}$ contains the static kinematic parameters of the robot.

This yields $3K$ non linear equations in the, say N, unknowns given by the robot kinematic parameters $\mathbf{x}$ and the unknown position of the tool top $\mathbf{p}$. Typically, we have $3K >> N$ similar to least squares problem. In this case, it is just a nonlinear least squares problem.

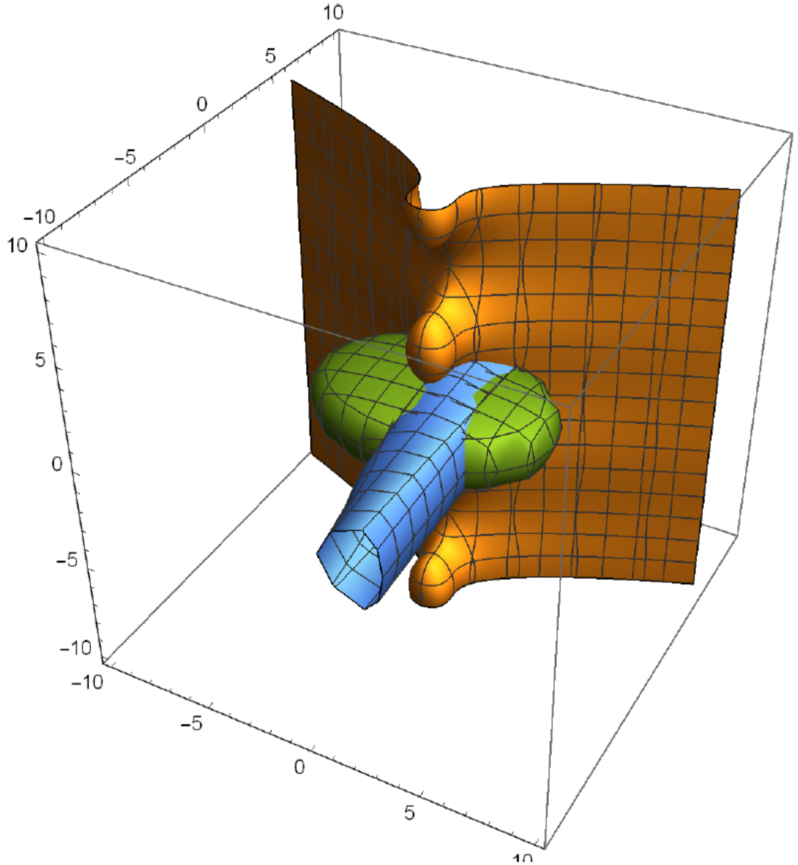
We have a good initial guess on $\mathbf{x}$ (robot factory settings). We can use Newtons method, where the Jacobian becomes $3K \times N$ dimensional. Solve the systems of linear equations in each step in Newtons method with SVD. Be aware of and handle singularity problems.

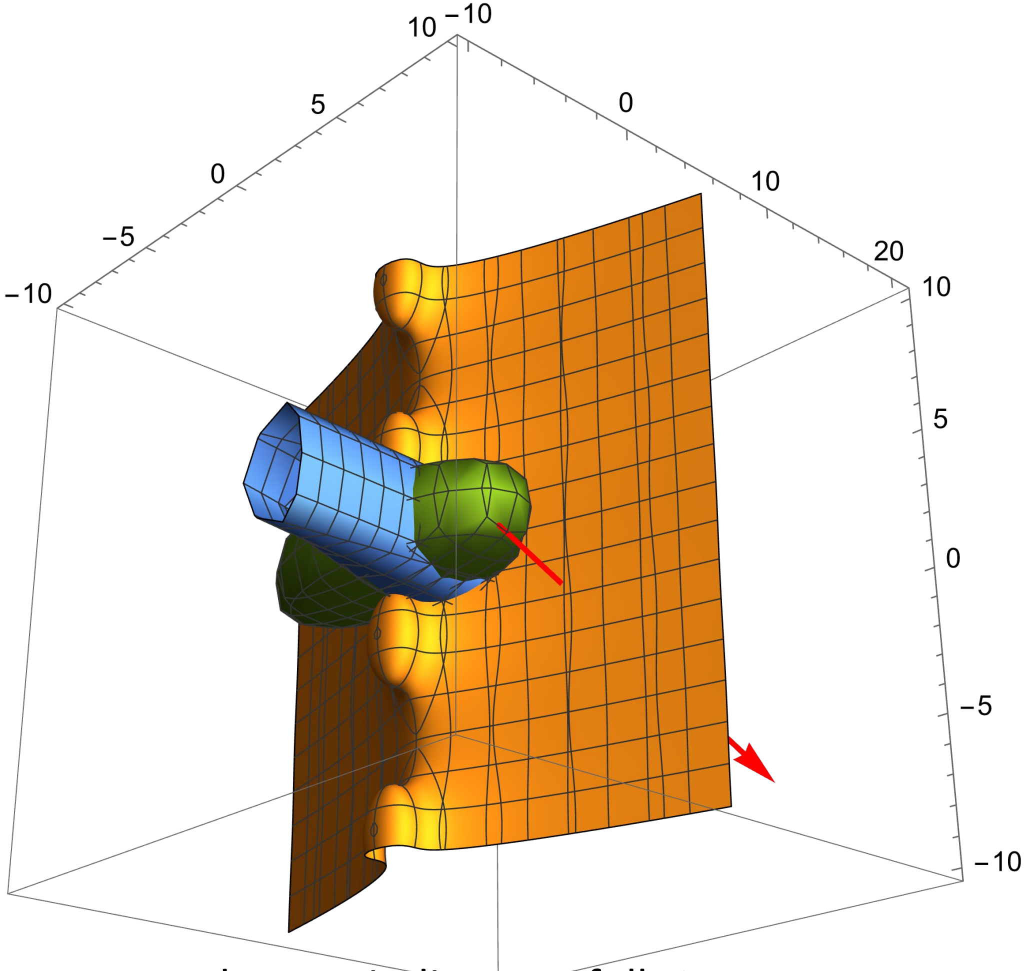
From now on: As many equations as unknowns, no singularity issues.

Control of Newton steps **far from the solution**

In[13]:=
```
ContourPlot3D[{f1[x1, x2, x3] == 0, f2[x1, x2, x3] == 0, f3[x1, x2, x3] ==
  {x1, -10, 10}, {x2, -10, 10}, {x3, -10, 10}]
```

Out[13]=





Red arrow indicates a full Newton step starting from (x1,x2,x3)=(5,-0.5,-1)

From now on: As many equations as unknowns, no singularity issues.

Control of Newton steps **far from the solution**

$$\mathbf{J}(\mathbf{x})\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}) \qquad (9.6.6)$$

Matrix equation (9.6.6) can be solved by LU decomposition as described in §2.3. The corrections are then added to the solution vector,

Notice: $\mathbf{x}_{old} \equiv \mathbf{x}$ in (9.6.6)

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x} \qquad (9.6.7)$$

May get worse from one step to the next...

Stepsize control to ensure "convergence: $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \lambda\,\delta\mathbf{x} \qquad 0 < \lambda \le 1$

How can we monitor if it "gets worse" for example here ? $\mathbf{F}(\mathbf{x}^{\text{old}}) = \begin{pmatrix} 0.013 \\ -0.02 \\ 0.04 \end{pmatrix} \quad \mathbf{F}(\mathbf{x}^{\text{new}}) = \begin{pmatrix} 0.02 \\ -0.015 \\ 0.035 \end{pmatrix}$

”Globally convergent” Newton

Define: $f(\mathbf{x}) \equiv \frac{1}{2}\mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x})$

$f(\mathbf{x}) \equiv \frac{1}{2}\mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x})$ implies that $\nabla f(\mathbf{x}) = \mathbf{F}(\mathbf{x})\mathbf{J}(\mathbf{x})$

We rewrite the Newton step as $\delta\mathbf{x} = -\mathbf{J}(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x})$

Directional derivative in the Newton step direction:

$$\nabla f(\mathbf{x}) \cdot \delta\mathbf{x} = -\mathbf{F}(\mathbf{x})\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x}) = -\mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x}) = -2f(\mathbf{x}) < 0$$

The Newton step is hence a descent direction of $f(\mathbf{x})$

$$\mathbf{F}(\mathbf{x}^{\mathrm{old}}) = \begin{pmatrix} 0.013 \\ -0.02 \\ 0.04 \end{pmatrix} \quad \mathbf{F}(\mathbf{x}^{\mathrm{new}}) = \begin{pmatrix} 0.02 \\ -0.015 \\ 0.035 \end{pmatrix}$$

$$f(\mathbf{x}^{\mathrm{old}}) = \frac{1}{2}(0.013^2 + 0.02^2 + 0.04^2) \simeq 0.0011$$

$$f(\mathbf{x}^{\mathrm{new}}) = \frac{1}{2}(0.02^2 + 0.015^2 + 0.035^2) \simeq 0.00093$$

Backtracking:

Newton step $\mathbf{p}$ Notice: We adopt NR notation, where until now $\mathbf{p}$ was called $\delta\mathbf{x}$.

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \lambda\mathbf{p}, \qquad 0 < \lambda \leq 1$$

The aim is to find λ so that $f(\mathbf{x}_{\text{old}} + \lambda\mathbf{p})$ has decreased sufficiently.

What should the criterion for accepting a step be?

A simple way to fix the first problem is to require the *average* rate of decrease of f to be at least some fraction α of the *initial* rate of decrease $\nabla f \cdot \mathbf{p}$:

$$f(\mathbf{x}_{\text{new}}) \leq f(\mathbf{x}_{\text{old}}) + \alpha \nabla f \cdot (\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}}) \tag{9.7.7}$$

Here the parameter α satisfies $0 < \alpha < 1$. We can get away with quite small values of α; $\alpha = 10^{-4}$ is a good choice.

From previous iteration, we have $f(\mathbf{x}_{\text{old}}) = \frac{1}{2}\mathbf{F}(\mathbf{x}_{\text{old}}) \cdot \mathbf{F}(\mathbf{x}_{\text{old}})$

Perform Newton step and compute $f(\mathbf{x}_{\text{old}} + \mathbf{p}) = \frac{1}{2}\mathbf{F}(\mathbf{x}_{\text{old}} + \mathbf{p}) \cdot \mathbf{F}(\mathbf{x}_{\text{old}} + \mathbf{p})$

Define $g(\lambda) \equiv f(\mathbf{x}_{\text{old}} + \lambda\mathbf{p})$

$g'(\lambda) = \nabla f(\mathbf{x}_{\text{old}} + \lambda\mathbf{p}) \cdot \mathbf{p}$

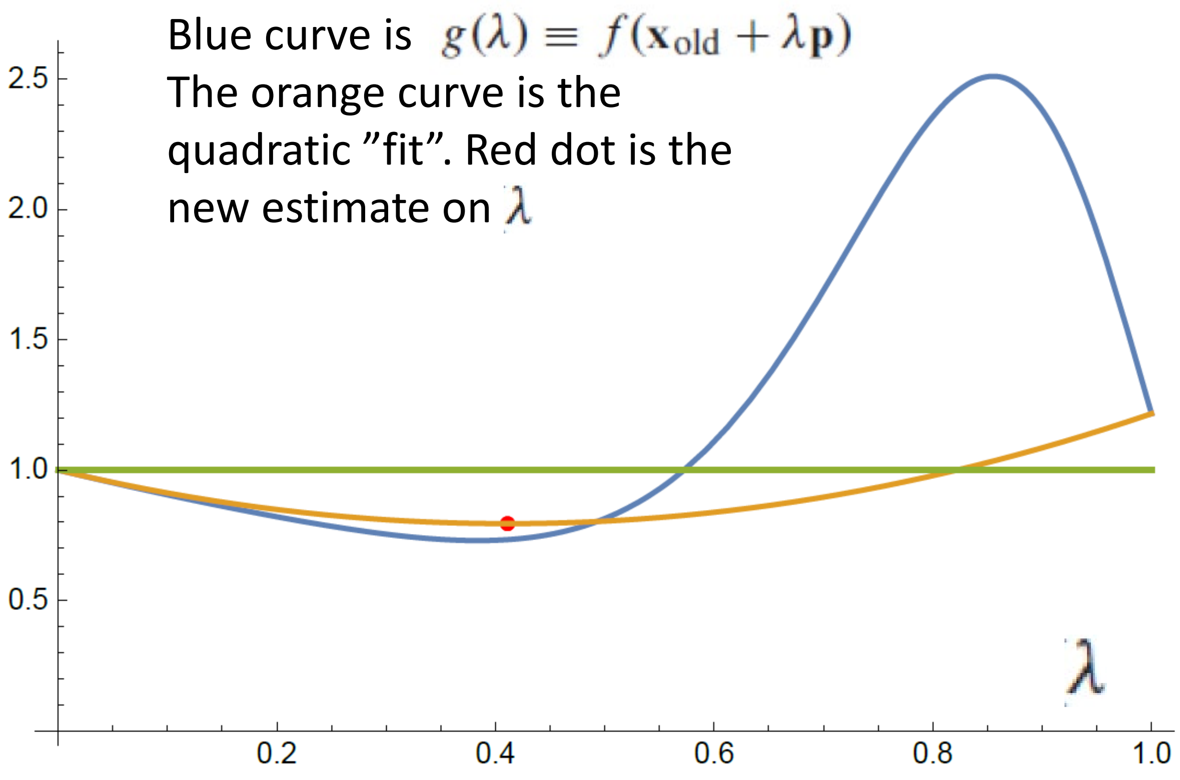
$g(0) \equiv f(\mathbf{x}_{\text{old}})$ $\qquad g(1) \equiv f(\mathbf{x}_{\text{old}} + \mathbf{p})$

$g'(0) \equiv \nabla f(\mathbf{x}_{\text{old}}) \cdot \mathbf{p} = -2f(\mathbf{x}_{\text{old}})$ All known

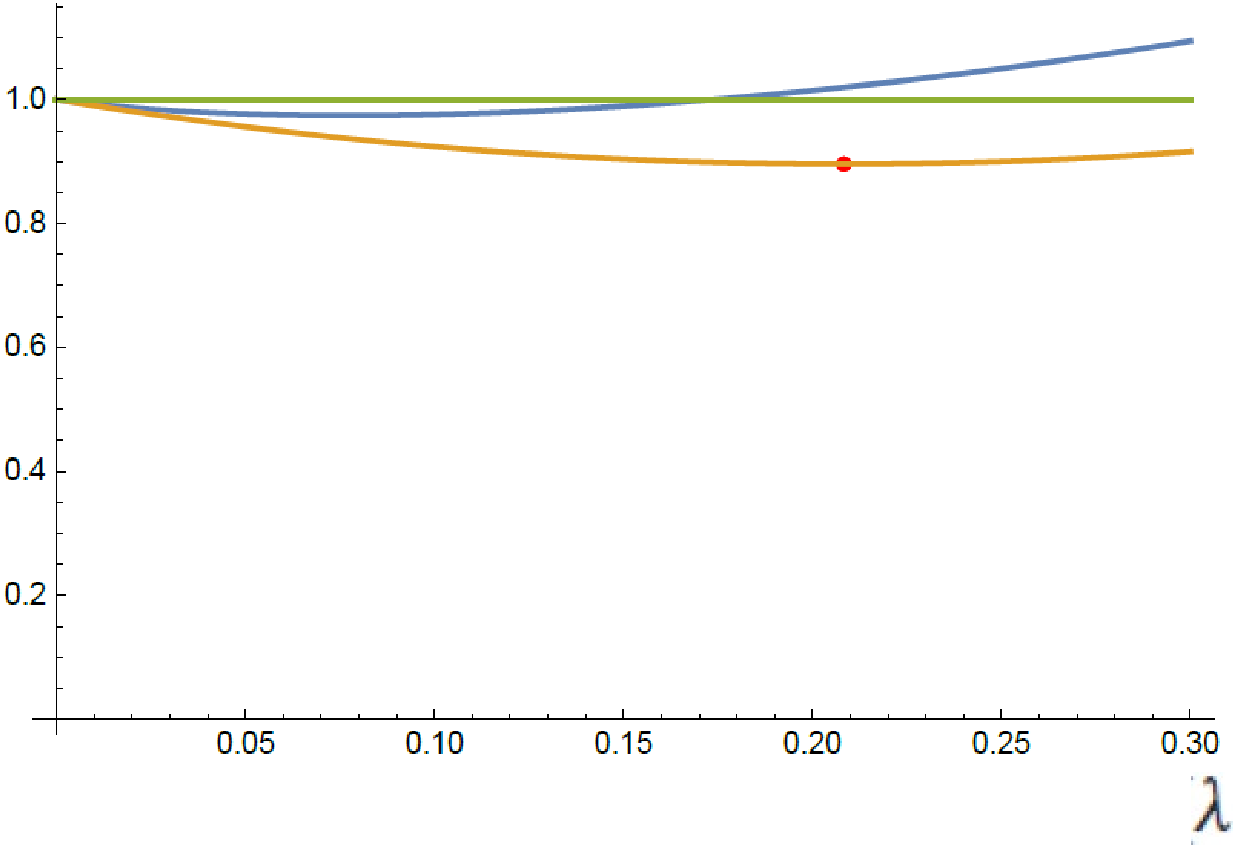
Approximate $g(\lambda)$ with a quadratic function: $g(\lambda) \approx [g(1) - g(0) - g'(0)]\lambda^2 + g'(0)\lambda + g(0)$

Minimum at $$\lambda = -\frac{g'(0)}{2[g(1) - g(0) - g'(0)]}$$

Stopping criterion: $g(\lambda) \equiv f(\mathbf{x}_{\text{old}} + \lambda\mathbf{p}) \leq f(\mathbf{x}_{\text{old}}) + \alpha\nabla f(\mathbf{x}_{\text{old}}) \cdot \lambda\mathbf{p} \equiv f(\mathbf{x}_{\text{old}})(1 - 2\lambda\alpha)$



Here $g(\lambda)$ is OK for the new λ



Here $g(\lambda)$ is not OK for the new λ

If stopping criterion is not satisfied:

On second and subsequent backtracks, we model g as a cubic in λ, using the previous value $g(\lambda_1)$ and the second most recent value $g(\lambda_2)$:

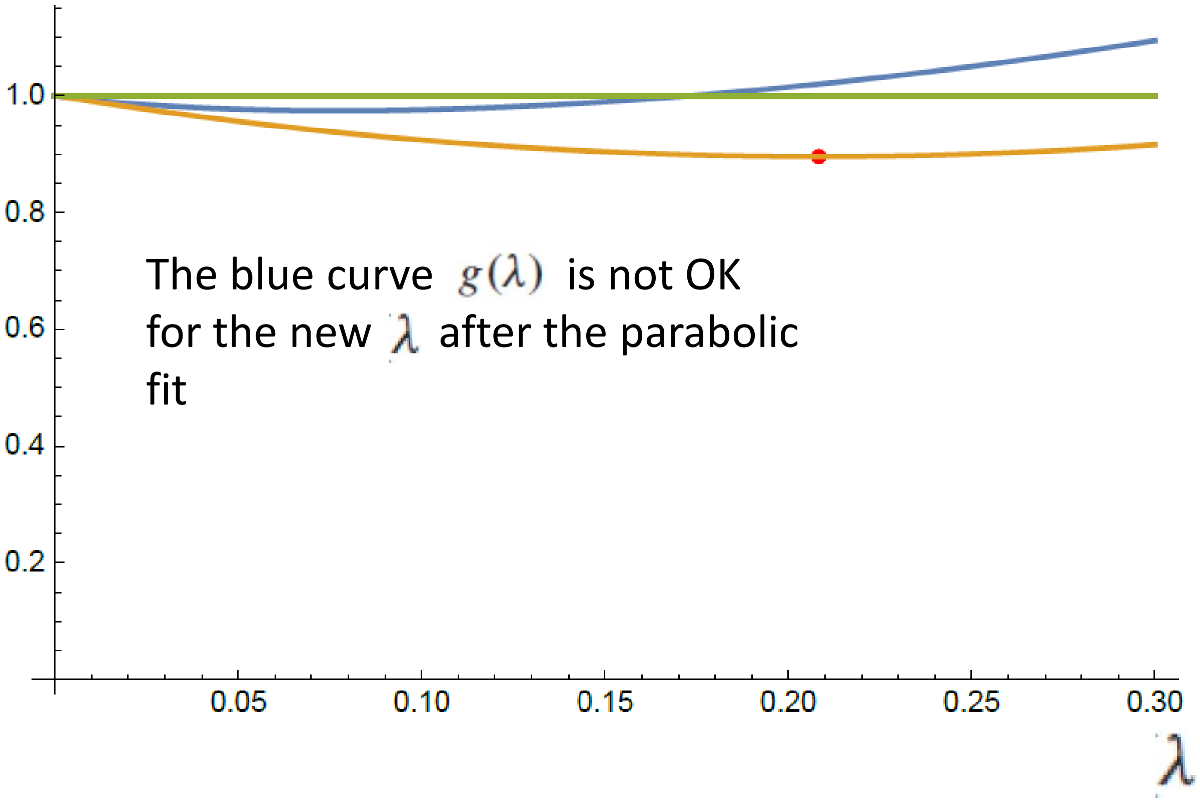
$$g(\lambda) = a\lambda^3 + b\lambda^2 + g'(0)\lambda + g(0) \tag{9.7.12}$$

The minimum of the cubic (9.7.12) is at

$$\lambda = \frac{-b + \sqrt{b^2 - 3ag'(0)}}{3a} \tag{9.7.14}$$

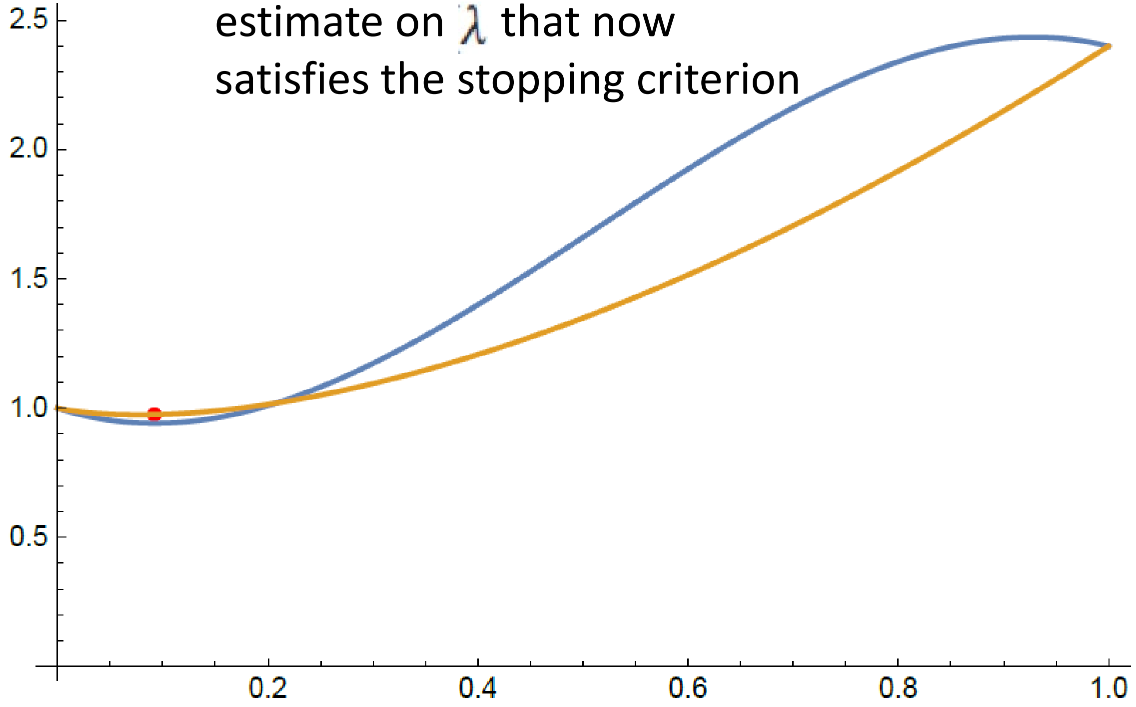
where
$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1/\lambda_1^2 & -1/\lambda_2^2 \\ -\lambda_2/\lambda_1^2 & \lambda_1/\lambda_2^2 \end{bmatrix} \cdot \begin{bmatrix} g(\lambda_1) - g'(0)\lambda_1 - g(0) \\ g(\lambda_2) - g'(0)\lambda_2 - g(0) \end{bmatrix}$$

Stopping criterion still: $g(\lambda) \equiv f(\mathbf{x}_{\text{old}} + \lambda \mathbf{p}) \le f(\mathbf{x}_{\text{old}}) + \alpha \nabla f(\mathbf{x}_{\text{old}}) \cdot \lambda \mathbf{p} \equiv f(\mathbf{x}_{\text{old}})(1 - 2\lambda\alpha)$



The blue curve $g(\lambda)$ is not OK for the new λ after the parabolic fit

Blue curve is $g(\lambda) \equiv f(\mathbf{x}_{\text{old}} + \lambda \mathbf{p})$
The orange curve is the cubic ”fit”. Red dot is the new estimate on λ that now satisfies the stopping criterion



Newton's method with backtracking: Jacobian

User input: A method to compute F(x) for a given input x.

Jacobian computation (all implemented in "newt", but relevant in many situations):

Already computed: $\mathbf{F}(\mathbf{x})$

Then compute $\mathbf{F}(\mathbf{x} + h\mathbf{e}_j)$ where $\mathbf{e}_j = (0, \ldots, 0, 1, 0, \ldots 0)$ where the 1 is at the j'th coordinate.

Compute the elements j'th column of the Jacobian as

$$\mathbf{J}_{ij} = \frac{[\mathbf{F}(\mathbf{x} + h\mathbf{e}_j)]_i - [\mathbf{F}(\mathbf{x})]_i}{h}; \quad i = 0, \ldots, N-1$$

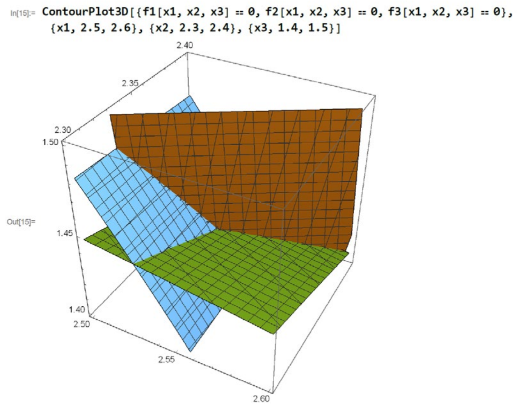
In total N extra F computations to compute the Jacobian.

NOTICE: There are special cases where most of the elements in the Jacobian are known to be zero. Remember to exploit that to not perform unnecessary F-computations.

Discussion and error estimating:

Notice that globally convergent Newton guarantees convergence to a local minima of $f(\mathbf{x}) \equiv \frac{1}{2}\mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x})$. However unless f(x)=0 at the minima, we have not found a solution.

Close to a solution, backtracking will not be necessary and globally convergent Newton becomes classical Newton. Then the error analysis from 1D applies with numerical values replaced by vector norms:



Method	Expected Order	Estimate of C	Estimate of $\lvert\epsilon_k\rvert$
Newton	2	$\dfrac{\lVert d_k\rVert}{\lVert d_{k-1}\rVert^2}$	$C\lVert d_k\rVert^2$ or $\lVert d_k\rVert$

Exam August 2023

Exercise 2 (20 points)

Consider the equations

$$\begin{aligned} x_0 + 2\sin(x_1 - x_0) - \exp(-\sin(x_1 + x_0)) &\equiv 0 \\ x_0\cos(x_1) + \sin(x_0) - 1 &\equiv 0 \end{aligned}$$

i) (3 points) With $x_0 = 1$ and $x_1 = 1$, state (with at least 6 digits) the values of the left hand sides of the two equations. (HINT: you should get approximately 0.597 and 0.382 respectively). Submit the used code.

ii) (4 points) State which methods from the course you can apply for this problem.

iii) (6 points) Perform 6 iterations with a method from the course using $x_0 = 1$ and $x_1 = 2$ as the initial guess and state the values of x_0 and x_1 after each of the 6 iterations. Submit the used code.

iv) (7 points) Provide an estimate of the error on the solution after 6 iterations. State the result and state a detailed explanation on how you arrived at the result

Mid term evaluation – lectures (Henrik)

Mid term evaluation – exercices (Jens)