Summary: Optimization with Constraints

\[
\min_{x} f(x) \quad \text{such that} \quad \begin{align*}
&c_m(x) - s_m = 0 \\
&s_m \geq 0 \quad m = 1 \ldots N_{\text{inequalities}} \\
&\min_{x} f(x) + \xi (c-s)^2 \\
&\text{penalty method, second term } \xi (c-s)^2 \text{ is optional}
\end{align*}
\]

KKT conditions: at constrained (local) minimum:

Augmented Lagrangian

\[
\nabla_x f - \sum_m (\lambda_m \nabla x c_m) = 0 \rightarrow
\]

\[
\begin{align*}
\lambda_m c_m &= 0 \\
\left[ \begin{array}{c} c \\ \lambda \\ s \end{array} \right] &\rightarrow \\
\begin{bmatrix}
\nabla f - \sum \lambda_m \nabla c_m \\
- \lambda_m c_m \\
- s
\end{bmatrix} &\rightarrow \\
F(x) = 0 = \\
\begin{bmatrix}
f(x) \\
\lambda_m c_m \\
- s
\end{bmatrix}
\end{align*}
\]

Newton \rightarrow SQP

If everything is linear: \rightarrow \text{SIMPLEX} (i.e. many business problems)

\[
g(x) = 0 \rightarrow x_N = G(x_1, \ldots, x_{N-1})
\]

Unconstrained \rightarrow \text{trust region Newton-type BFGS}

gigantic \rightarrow \text{conjugate gradient}

In Chemical Engineering, the problems often involve models with differential equations:

\[
f(x) = \sum_i w_i \left( Y_i(t_i; x) - \hat{Y}_i(t_i; \hat{x}) \right)
\]

knobs \quad what we need
(can adjust) \quad what we produce

Need Jacobian of G with respect to Y; need in stiff solver to solve.

\[
\frac{dY}{dt} = G(Y; \hat{x}) \quad Y(t_0) = Y_0(x)
\]
Need gradient and f.

To use all of our methods, we need to be able to compute:

\[
\frac{\partial f}{\partial x_j} = \sum_i w_i \left( \frac{\partial Y_{\sigma,i}}{\partial x_j} - \frac{\partial Y_i(t_f)}{\partial x_j} \right)
\]

how do you compute this?

\[
\frac{\partial}{\partial x_j} \left( \frac{\partial Y_i}{\partial t} \right) = \left( \sum_n \frac{\partial G_i}{\partial Y_n} \frac{\partial Y_n}{\partial x_j} \right) + \frac{\partial G_i}{\partial x_j}
\]

chain rule

\[
\frac{\partial}{\partial t} \left( \frac{\partial Y_i}{\partial x_j} \right)
\]

solve this with initial conditions

\[
\begin{cases}
\mathbf{s}_{ij} = 0 & \text{most knobs} \\
\mathbf{s}_{ij} = 1 & \text{for adjustment of } Y_0
\end{cases}
\]

Initial Conditions

What is \( s_{ij}(t_0) \)?

\[
\begin{align*}
\mathbf{s}_{ij}(t_0) &= 0 & \text{most knobs} \\
\mathbf{s}_{ij}(t_0) &= 1 & \text{for adjustment of } Y_0
\end{align*}
\]

Professor Barton teaches an advanced course in optimization.
Boundary Value Problems (BVPs)

Conservation Laws: \( \frac{\partial \phi}{\partial t} = -\nabla \cdot (\phi \mathbf{v}) - \nabla \cdot \mathbf{J}_D + S(\phi) \)

\( \mathbf{J}_D = \mathbf{v} + \nabla \phi \)

isotropic: \( \mathbf{J}_D = -c \nabla \phi \)

for steady-state, isotropic: \( 0 = -\nabla \cdot (\phi \mathbf{v}) - c \nabla^2 \phi + S(\phi) \quad \forall \mathbf{x} \) (Laplacian)

Boundary conditions:

Dirichlet \( \phi(\text{boundary}) = \text{number} \)

von Neumann \( \nabla \phi(\text{boundary}) = \text{number or 0} \)

Symmetry \( \frac{\partial \phi}{\partial x_j} = 0 \)

\( \phi(x) \) infinite \{rare to find exact\}

\( \phi_{\text{approx}}(x) = f(x; \xi) \) \quad \text{adjust: large finite number (10^4)}

Basis function expansions \( \phi_{\text{approx}} = \sum_{n=1}^{N_{\text{basis}}} c_n \Psi_n(x) \)

\( \int_{m=1, N_{\text{basis}}}^{m \neq C.B.} \Psi_m(x) \left( -\nabla \cdot (\phi_{\text{approx}} \mathbf{v}) + c \nabla^2 \phi_{\text{approx}} + S(\phi_{\text{approx}}) \right) = 0 \)

\( \phi_{\text{approx}}(x) = f(x; \{ \phi_i \}) \quad \phi_{\text{approx}}(x_i) \quad \{ x_i \} = \text{mesh grid} \)

some interpolation called “Residual”

Finite difference approximation to differential equation

Dirichlet \( \phi(\text{boundary}) \) \( \{ \phi \} \quad i = 1, N \)

\( \left. \frac{\partial \phi}{\partial x} \right|_{x_i} = \frac{\phi_{i+1} - \phi_{i-1}}{x_{i+1} - x_{i-1}} \quad \phi_{B.C.} \)

von Neumann
\[ \frac{\partial \phi}{\partial x} \bigg|_{x_0} \text{ given } \phi_0? \quad \text{Usual } \rightarrow \text{ 2nd order polynomials} \]

\[ \phi(x) = \phi(x_0) + \frac{\partial \phi}{\partial x} \bigg|_{x_0} (x - x_0) + \frac{1}{2} \frac{\partial^2 \phi}{\partial x^2} \bigg|_{x_0} (x - x_0)^2 \]

\[ \phi_0 = f(\phi_1, \phi_2) \quad \frac{\partial \phi}{\partial x} \bigg|_{x_1} = \frac{\phi_2 - f(\phi_1, \phi_2)}{x_2 - x_0} \]

\[ \phi(x_1) = \phi_0 + \frac{\partial \phi}{\partial x} \bigg|_{x_0} (x_1 - x_0) + \frac{1}{2} \frac{\partial^2 \phi}{\partial x^2} \bigg|_{x_0} (x_1 - x_0)^2 \ldots \]

\[ \phi(x_2) = \phi_0 + \frac{\partial \phi}{\partial x} \bigg|_{x_0} (x_2 - x_0) + \frac{1}{2} \frac{\partial^2 \phi}{\partial x^2} \bigg|_{x_0} (x_2 - x_0)^2 \ldots \]

\[ \phi_0 = \frac{\phi_1 (x_2 - x_0)^2 - \phi_2}{(x_2 - x_0)^2 - (x_1 - x_0)^2 - 1} \quad \text{for } \frac{\partial \phi}{\partial x} \bigg|_{x_0} = 0 \]

If \( \Delta x \) uniform, \( \phi_0 = \frac{4\phi_1 - \phi_2}{3} \)

This is how you find out B.C. with second order polynomial schemes and a finite difference approximation.