Nonlinear Systems
Part II Roadmap

- Part I – Linear Algebra (units 1-12) \( Ac = b \)
  - (units 13-16) Optimization -> Nonlinear Equations -> 1D roots/minima
  - (units 17-18) Computing/Avoiding Derivatives
  - (unit 19) Hack 1.0: “I give up” \( H = I \) and \( J \) is mostly 0 (descent methods)
  - (unit 20) Hack 2.0: ”It’s an ODE!?“ (adaptive learning rate and momentum)
Recall: Jacobian

- Given \( F(c) = \begin{pmatrix} F_1(c) \\ F_2(c) \\ \vdots \\ F_m(c) \end{pmatrix} \) the Jacobian of \( F(c) \) has entries \( J_{ij} = \frac{\partial F_i}{\partial c_j}(c) \)

- Thus, the Jacobian \( J(c) = F'(c) = \begin{pmatrix} \frac{\partial F_1}{\partial c_1}(c) & \frac{\partial F_1}{\partial c_2}(c) & \cdots & \frac{\partial F_1}{\partial c_n}(c) \\ \frac{\partial F_2}{\partial c_1}(c) & \frac{\partial F_2}{\partial c_2}(c) & \cdots & \frac{\partial F_2}{\partial c_n}(c) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial c_1}(c) & \frac{\partial F_m}{\partial c_2}(c) & \cdots & \frac{\partial F_m}{\partial c_n}(c) \end{pmatrix} \)
Linearization

- Solving the nonlinear system $F(c) = 0$ is difficult.
- It can be linearized by considering the first term in the multidimensional version of the Taylor expansion: $F(c) \approx F(c^*) + F'(c)(c - c^*) = F(c^*) + F'(c)\Delta c$
  - This is more valid when $\Delta c$ is small (i.e. for $c$ close enough to $c^*$).
  - This can be alternatively written as $F(c) - F(c^*) \approx F'(c)\Delta c$.
- The chain rule $\frac{dF(c)}{dt} = F'(c) \frac{dc}{dt}$ can be written in differential form for vanishingly small differentials $dF(c) = F'(c)dc$
  - This is often referred to as the total derivative.
  - Using finite size differentials leads to the approximation $\Delta F(c) \approx F'(c)\Delta c$.
- In 1D, $df = f'(c)dc$ and $\Delta f \approx f'(c)\Delta c$ are the usual $\frac{df}{dc} = f'(c)$ and $\frac{\Delta f}{\Delta c} \approx f'(c)$.
Newton’s Method

• Iteratively, starting with $c^0$, recursively find: $c^1, c^2, c^3, ...$

• Newton’s Method uses $\Delta F(c) \approx F'(c)\Delta c$ to write $F(c^{q+1}) - F(c^q) = F'(c^q)\Delta c^q$ where $\Delta c^q = c^{q+1} - c^q$
  • Aiming for $F(c) = 0$ motivates setting $F(c^{q+1}) = 0$
  • Alternatively, one could set $F(c^{q+1}) = \beta F(c^q)$ where $0 \leq \beta < 1$ gives a rate at which one aims to shrink $F(c^q)$ towards zero
  • This gives $F'(c^q)\Delta c^q = (\beta - 1)F(c^q)$ with $\beta$ often set to 0

• The linear system $F'(c^q)\Delta c^q = (\beta - 1)F(c^q)$ is solved for $\Delta c^q$, which is used to update $c^{q+1} = c^q + \Delta c^q$
Newton’s Method

• Requires repeated solving of a linear system, which is one reason why robustness and efficiency for linear system solvers is so important
  • Need to consider size, rank, conditioning, symmetry, etc. of $F'(c^q)$

$F'(c^q)$ may be difficult to compute, since it requires every first derivative
  • In particular, Newton’s Method contains linearization errors, so useful approximations of $F'(c^q)$ seem valid/worthwhile (e.g. symmetrization, etc.)
  • This is discussed more in units 17/18

• Generally, there are no guarantees on convergence
  • May converge to any one of many roots when multiple roots exist or not converge at all
Linear System Solvers (Review)

• Theory, all matrices: **SVD** (unit 3/9/11)
• Square, full rank, dense:
  • LU factorization with pivoting (unit 2)
  • SPD: **Cholesky** factorization (unit 4), **Symmetric approximation** (unit 4)
• Square, full rank, sparse (iterative solvers) (unit 5):
  • SPD (sometimes SPSD): **Conjugate Gradients**
  • Nonsymmetric/Indefinite: GMRES, MINRES, BiCGSTAB (not steepest descent)
• Tall, full rank (least squares to minimize residual) (unit 8):
  • normal equations (unit 9/10), **QR**, Gram-Schmidt, **Householder** (unit 10)
• Any size/rank (minimum norm solution) (unit 11):
  • **Pseudo-Inverse**, PCA approximation, **Power Method** (unit 11)
  • Levenberg-Marquardt (iteration too), Column Space Geometric Approach (unit 12)
Line Search

- Given the linearization error in $F'(c^q)\Delta c^q = -F(c^q)$, the resulting $\Delta c^q$ often leads to a poor estimate for $c^{q+1}$ via $c^{q+1} = c^q + \Delta c^q$

- Thus, $\Delta c^q$ is often merely treated as a search direction, i.e. $c^{q+1} = c^q + \alpha^q \Delta c^q$

- The parameterized line $c^{q+1}(\alpha) = c^q + \alpha \Delta c^q$ is used as a 1D (input) domain

- Find $\alpha$ such that $F(c^{q+1}(\alpha)) = 0$ simultaneously for all equations

- Safe Set methods restrict $\alpha$ in various ways, e.g. $0 \leq \alpha \leq 1$
Line Search

- Since $F$ is vector valued, consider $g(\alpha) = F\left(c^{q+1}(\alpha)\right)^T F\left(c^{q+1}(\alpha)\right) = 0$
- Note $g(\alpha) = F\left(c^{q+1}(\alpha)\right)^T F\left(c^{q+1}(\alpha)\right) \geq 0$, so solutions to $F\left(c^{q+1}(\alpha)\right) = 0$ are minima of $g(\alpha)$ (and more difficult to find)
- $g(\alpha)$ might be strictly positive, but minimizing $g(\alpha)$ can help to make progress towards a solution

- **Option 1**: find simultaneous roots of the vector valued $F\left(c^{q+1}(\alpha)\right) = 0$
- **Option 2**: find roots or minimize $g(\alpha) = \frac{1}{2} F^T \left(c^{q+1}(\alpha)\right) F\left(c^{q+1}(\alpha)\right)$
Optimization Problems

- Minimize the scalar cost function $\hat{f}(c)$ by finding the critical points where 
  $\nabla \hat{f}(c) = J_f^T(c) = F(c) = 0$

- $F'(c^q)\Delta c^q = -F(c^q)$ gives the search direction, where $F'(c) = J_F(c) = H_f^T(c)$

- That is, solve $H_f^T(c^q)\Delta c^q = -J_f^T(c^q)$ to find the search direction $\Delta c^q$

- **Option 1**: find simultaneous roots of the vector valued $J_f^T(c^{q+1}(\alpha)) = 0$ which are the critical points of $\hat{f}(c)$

- **Option 2**: find roots or minimize $g(\alpha) = \frac{1}{2}J_f^T(c^{q+1}(\alpha))J_f^T(c^{q+1}(\alpha))$ to find or make progress toward critical points of $\hat{f}(c)$

- **Option 3**: minimize $\hat{f}(c^{q+1}(\alpha))$ directly