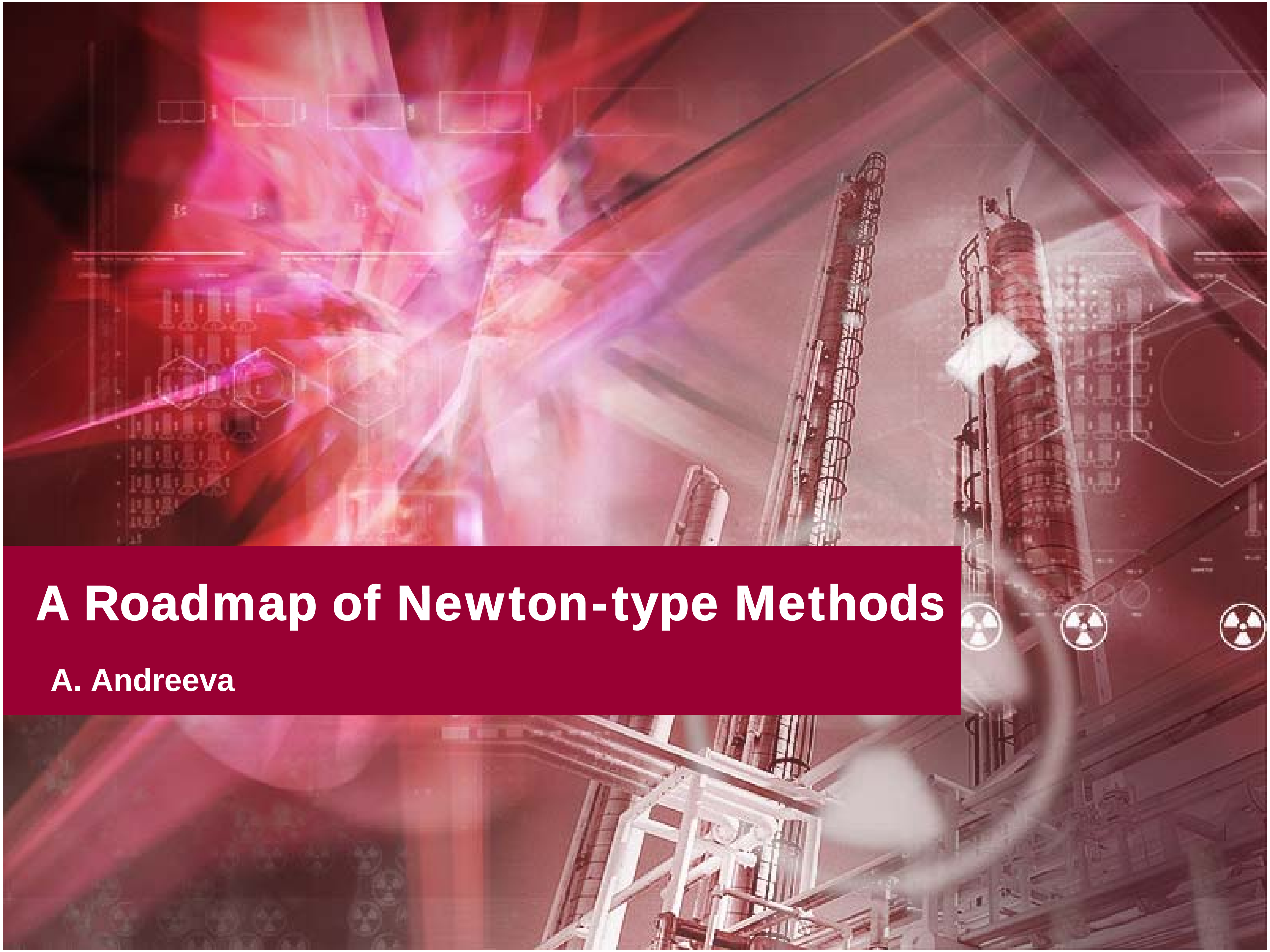

A Roadmap of Newton-type Methods

A. Andreeva

Ordinary Newton method

- For general nonlinear problems:

$$F'(x^k)\Delta x^k = -F(x^k), \;\; x^{k+1} = x^k + \Delta x^k$$

- For system of *n* nonlinear equations Jacobian matrix is required
- First we compute the Newton corrections Δx^k and then improve the iterates x^k to obtain x^{k+1}

Simplified Newton method

- Keeps the initial derivative throughout the whole iteration:

$$F'(x^0)\overline{\Delta x}^k = -F(x^k), \quad x^{k+1} = x^k + \overline{\Delta x}^k$$

- Saves computational cost per iteration

Newton-like methods

- In finite dimension, the Jacobian matrix is:
  - Replaced by some fixed 'close by' Jacobian $F'(z),\ z \neq x^0$
  - Approximate $F'(x^k)$ by $M(x^k)$

$$M(x^k)\,\delta x^k = -F(x^k), \quad x^{k+1} = x^k + \delta x^k$$

Exact Newton methods

- When the equation

$$F'(x^k)\Delta x^k = -F(x^k)$$

  can be solved using *direct* elimination methods, we speak of *exact* Newton methods
- Erroneous when scaling issues are ignored

Local versus global Newton methods

- Local Newton methods require sufficiently good initial guess
- Global Newton methods compensate by virtue of damping or adaptive trust region strategies
- *Exact* global Newton codes:
  - NLEQ-RES – residual based
  - NLEQ-ERR – error oriented
  - NLEQ-OPT – convex optimization

Inexact Newton methods

- Inner iteration $$F'(x^k)\delta x_i^k = -F(x^k) + r_i^k$$ $$x_i^{k+1} = x^k + \delta x_i^k$$
- Outer iteration $$x^{k+1} = x_i^{k+1}$$
- In comparison with exact Newton methods an error arises: $\delta x^k - \Delta x^k$
- GIANT – Global Inexact Affine invariant Newton Techniques

Preconditioning

- Direct elimination of ‘similar’ linear systems

$$C_L F'(x^k)\, C_R C_R^{-1}(\delta x_i^k - \Delta x_i^k) = C_L r_i^k$$

- Residual or error norm need to be replaced by their preconditioned counterparts

$$\left\| r_i^k \right\|,\ \left\| \delta x_i^k - \Delta x_i^k \right\| \rightarrow \left\| C_L r_i^k \right\|,\ \left\| C_R^{-1}(\delta x_i^k - \Delta x_i^k) \right\|$$

Matrix-free Newton methods

- Numerical difference approximation

$$F'(x)v = \frac{F(x+\delta v) - F(x)}{\delta}$$

Secant methods

- Substitute the tangent by the secant

$$f'(x^k + \delta x_k) \rightarrow \frac{f(x^k + \delta x_k) - f(x^k)}{\delta x_k} = j_{k+1}$$

- Compute the correction

$$\delta x_{k+1} = -\frac{f(x^{k+1})}{j_{k+1}}, \qquad x^{k+1} = x^k + \delta x_k$$

- Converges locally *superlinearly*

Quasi-Newton methods

- Extends the secant idea to system of equations

$$J\delta x_{\kappa} = F(x^{k+1}) - F(x^{k})$$

- Previous quasi-Newton step: $J_k \delta x_k = -F(x^k)$

- Jacobian rank-1 update: $$J_{k+1} = J_k + \frac{F(x^{k+1})z^T}{z^T \delta x_k}$$

- Next quasi-Newton step: $J_{k+1}\delta x_{k+1} = -F(x^{k+1})$

Gauss-Newton methods

- Appropriate for nonlinear least square problems
- Must be statistically well-posed (to be discussed later in Sections 2 and 3)
- Two classes of Gauss-Newton methods:
  - Local – good initial guess is required
  - Global - otherwise

Quasilinearization

- Infinite dimensional Newton methods for operator equations
- The linearized equations can be solved only approximately
- Similar to inexact Newton methods, where the ‘truncation errors’ correspond to ‘approximation errors’

Inexact Newton multilevel methods

- Infinite dimensional linear Newton systems are approximately solved by linear multilevel methods
- When the approximation errors are controlled within an abstract framework of inexact Newton methods, we speak of *adaptive* Newton multilevel method

Multilevel Newton methods

- Schemes wherein a finite dimensional Newton multigrid method is applied on each level

Nonlinear multigrid methods

- Not Newton methods
- Rather fix point iteration methods
- Not treated here

Adaptive inner solver for inexact Newton Methods

- Idea: solve iteratively the linear systems for the Newton corrections
- The inexact Newton system is given as:

$$Ay_i = b - r_i, \qquad i = 0,1,\ldots,i_{max}$$

- Several termination criteria:
  - Residual norm $||r_i||$ is small enough
  - Error norm $||y - y_i||$ is small enough
  - Energy norm $||A^{1/2} (y - y_i)||$ of the error is small enough

Residual norm minimization: GMRES

- Initial approximation $y_0 \approx y$, initial residual $r_0 = b\text{-}Ay_0$,
- Set: $\beta = ||r_0||$, $v_1 = r_0 / \beta$, $V_1 = v_1$, iterate $i = 1,2,\ldots,i_{max}$
- Step1. Ortogonalization:

$$\hat{v}_{i+1} = Av_i - V_i h_i \quad where \quad h_i = V_i^T Av_i$$

- Step2. Normalisation:

$$v_{i+1} = \frac{\hat{v}_{i+1}}{\left\| \hat{v}_{i+1} \right\|_2}$$

Residual norm minimization: GMRES

- Step3. Update:

$$V_{i+1} = (V_i v_{i+1})$$

$$H_i = \begin{pmatrix} H_{i-1} & h_i \\ 0 & \|\hat{v}_{i+1}\|_2 \end{pmatrix}$$

for i=1 drop the left block column

- Step4. Least squares problem: $z_i = min \left\| \beta e_1 - H_i z \right\|$
- Step5. Approximate solution: $y_i = V_i z_i + y_0$

Characteristics of GMRES

- **Storage:** up to iteration i requires to store $i+2$ vectors of length n
- **Computational amount:** each iteration performs one matrix/vector multiplication. Up to iteration i, i^2n flops
- **Preconditioning:** best preconditioning for $C_L = I$

Energy norm minimization: PCG

- For symmetric positive definite matrix *A* the **energy product** and **energy norm** are defined as:

$$(u,v) = \langle u, Av \rangle \ \ and \ \ \|u\|_A^2 = (u,u)$$

- Idea: for positive definite $B \approx A^{-1}$ is much easier to compute $z = Bc$ then $Ay = b$.

Error norm minimization: CGNE

- Idea: minimize the norm $||y - y_i||$
- Initialize: initial approximation y_0, initial residual $r_0 = b – Ay_0$
- Set: $p_0 = 0, \beta_0 = 0, \sigma_0 = ||r_0||^2$

Error norm minimization: CGNE

$$For\ i = 1,2,\ldots,i_{max}$$

$$p_i = A^T r_{i-1} + \beta_{i-1} p_{i-1}$$

$$\alpha_i = \sigma_{i-1} / \|p_i\|^2$$

$$\gamma_{i-1}^2 = \alpha_i \sigma_{i-1} \qquad (\textit{Euclidean error contribution}\ \|y_i - y_{i-1}\|^2)$$

$$y_i = y_{i-1} + \alpha_i p_i, \qquad r_i = r_{i-1} - \alpha_i A p_i$$

$$\sigma_i = \|r_i\|^2, \qquad \beta_i = \sigma_i / \sigma_{i-1}$$

Characteristics of CGNE

- **Storage:** up to iteration *i* requires only 3 vectors of length *n*
- **Computational amount:** up to step *i* the Euclidean inner products sum up to *5in* flops
- **Preconditioning:** $C_R^{-1}(y - y_i)$ is minimized. Therefore, only left preconditioning should be realized.