Power Flow Analysis

STABILITY ANALYSIS OF NONLINEAR SYSTEMS (EEEN50100)

Prof. Federico Milano

Email: federico.milano@ucd.ie

Tel.: 01 716 1844

Room 157a – Engineering & Materials Science Centre

School of Electrical & Electronic Engineering

University College Dublin

Dublin, Ireland

Contents

- Maps
- Power Flow Problem
- Standard Power Flow Solution Methods
- Continuous Newton's Method

Maps

Definition and Needs of Maps

- A map is a function defined as follows:

$$\boldsymbol{x}_{n+1} = \boldsymbol{h}(\boldsymbol{x}_n), \quad n = 0, 1, 2, \ldots \tag{1}$$

- Maps are used to describe the dynamic behavior of discrete systems.
- In general, continuous systems have to be simulated and solved through (discrete) iterative methods, which, in turns, are a discrete representation of the original continuous system.

Example – Logistic Map (II)

- A typical example of discrete dynamical systems is the *logistic map*.
- The map was introduced in 1976 by the biologist Robert May as a discrete-time demographic model:

$$x_{n+1} = r x_n (1 - x_n), \quad n = 0, 1, 2, \ldots \tag{2}$$

where:

x_n is a number between zero and one, and represents the ratio of existing population to the maximum possible population at year n, and hence x_0 represents the initial ratio of population to max. population (at year 0).

r is a positive number, and represents a combined rate for reproduction and starvation.

- The logistic map is often cited as an example of how complex, chaotic behaviours can arise from very simple non-linear dynamical equations.

Example - Newton's Method

- Let consider the problem of finding a solution $\boldsymbol{y}_s$ of the equation $\boldsymbol{g}(\boldsymbol{y})$ that satifies:

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}_s) \tag{3}$$

- The Newton's method consists in iterating the following map:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n - \boldsymbol{g}_{\boldsymbol{y}}^{-1}(\boldsymbol{y}_n) \cdot \boldsymbol{g}(\boldsymbol{y}_n) = \boldsymbol{h}_{NR}(\boldsymbol{y}_n), \quad n = 0, 1, 2, \ldots \tag{4}$$

- Whether or not the map (3) is able to find the solution $\boldsymbol{y}_s$, i.e., whether $\lim_{n\to\infty} \boldsymbol{h}_{NR}(\boldsymbol{y}_n) \to \boldsymbol{y}_s$ depends on the convergence properties of map itself.

Fixed Points

- A fixed point $\tilde{\boldsymbol{x}}$ of a map $\boldsymbol{h}$ is defined as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{h}(\tilde{\boldsymbol{x}}) \tag{5}$$

- If $\boldsymbol{h} : \mathbb{R}^p \mapsto \mathbb{R}^p$ is smooth, and ρ are the eigenvalues of $\boldsymbol{h}_{\boldsymbol{x}}|_{\tilde{\boldsymbol{x}}}$, then, the fixed point is:
  - *stable* if $|\rho_i| < 1$, for all $i = 1, 2, \ldots, p$;
  - *unstable* if there is at least one eigenvalue such that $|\rho_i| > 1$;

- If one or more eigenvalues satisfy the condition $|\rho_i| = 1$, then more investigation is needed (possibly, the fixed point is a bifurcation point of the map).

Example - Linear Scalar Map

- Let consider a linear scalar map:

$$x_{n+1} = \mu x_n, \quad n = 0, 1, 2, \ldots \tag{6}$$

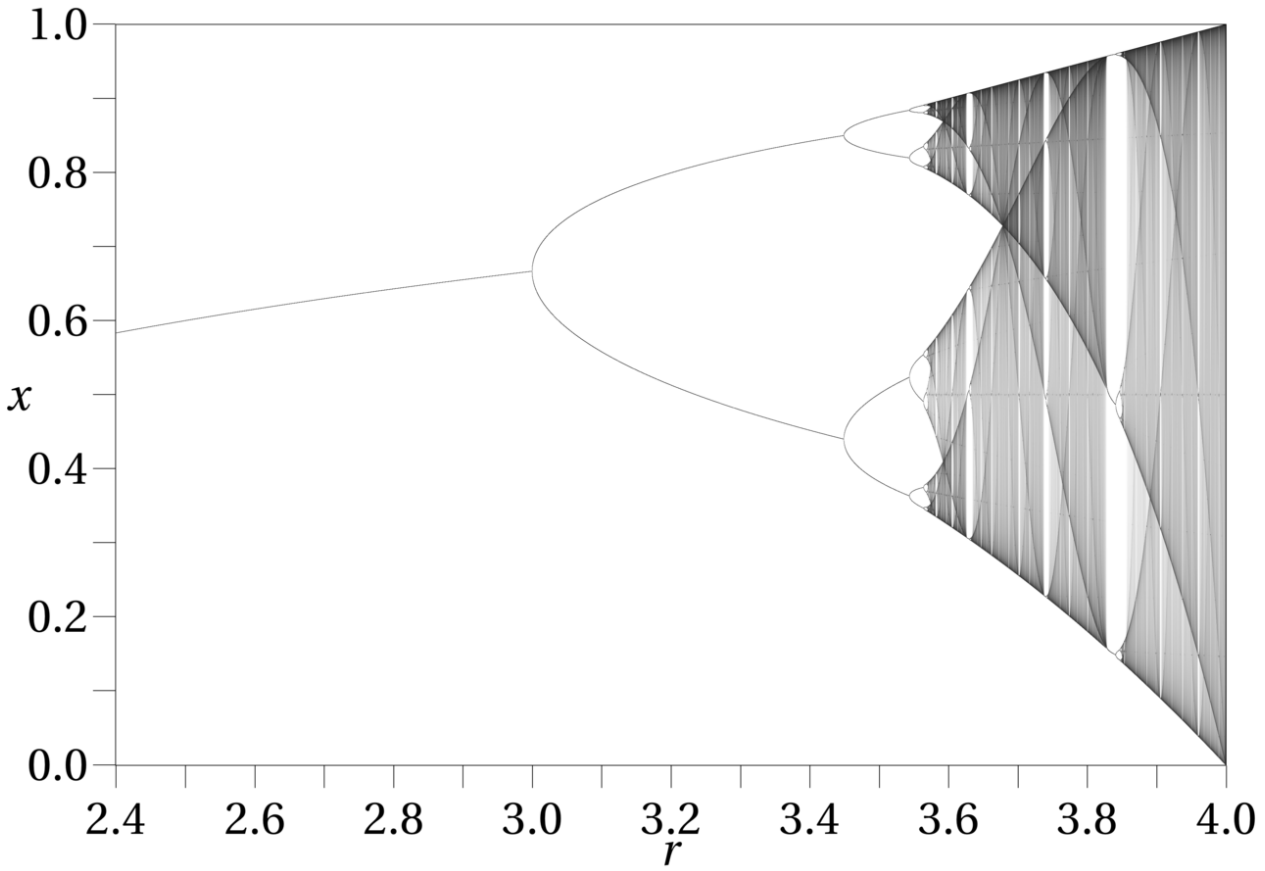
- In this case, we know the explicit solution:

$$x_{n+1} = \mu^n x_0 \tag{7}$$

- The origin $\tilde{x} = 0$ is a fixed point for (6).
- A small perturbation ϵ_0 around the fixed point $\tilde{x} = 0$ gives $\epsilon_{n+1} = \mu^n \epsilon_0$.
- The origin is asymptotically stable if $-1 < \mu < 1$, stable but not asymptotically stable if $\mu = 1$, and unstable if $|\mu| > 1$.

Example - Logistic Map (II)

- Despite its simplicity the bifurcation diagram of the logistic map is extremely complex.



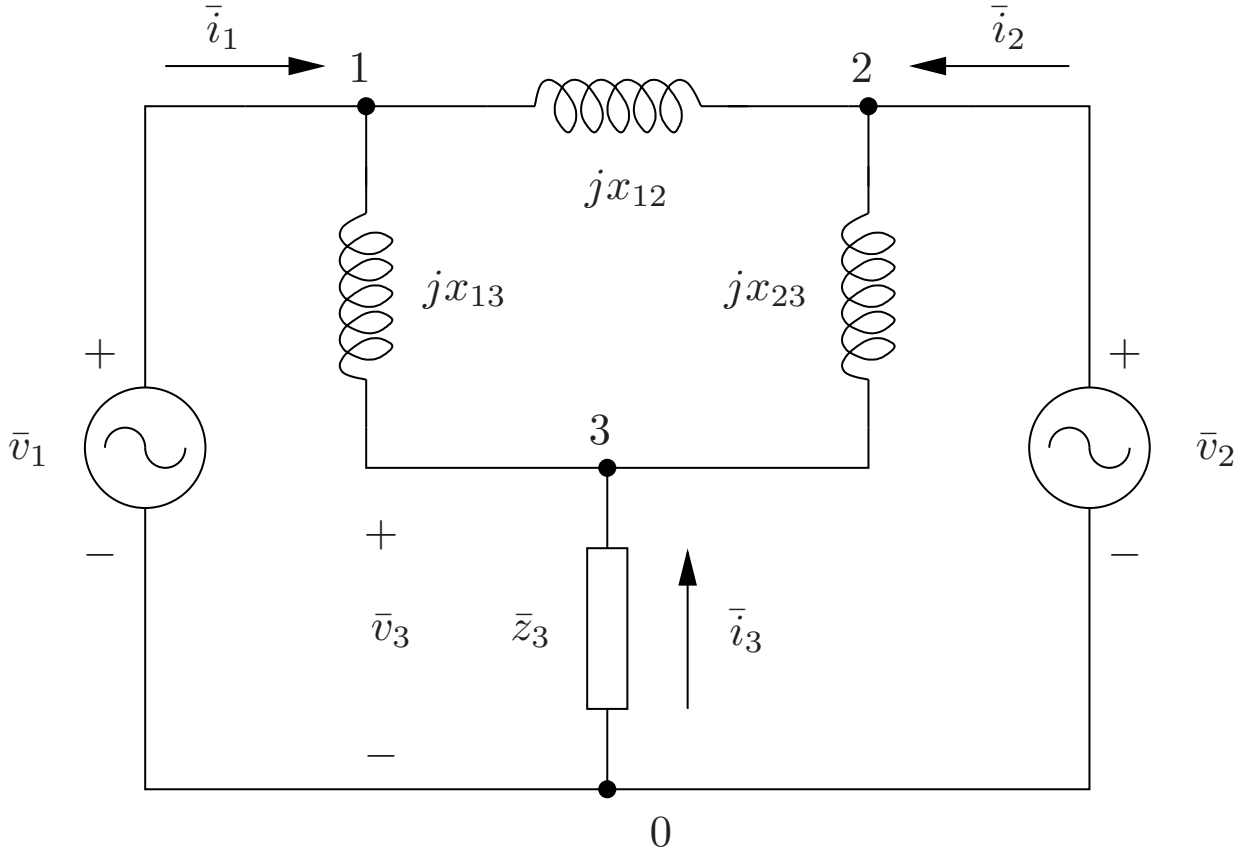
Power Flow Problem

Background (I)

- A classical problem of circuit theory is to find all branch currents and all node voltages of an assigned circuit.
- Typical input data are generator voltages as well as the impedances of all branches.
- If all impedances are constant, the resulting set of equations that describe the circuit is **linear**.

Example 1 (I)

- Classical circuit problem.



Example 1 (II)

- Using the branch current method, one has:

$$
\begin{aligned}
0 &= \frac{\bar{v}_1 - \bar{v}_2}{jx_{12}} + \frac{\bar{v}_1 - \bar{v}_3}{jx_{13}} - \bar{i}_1 \\
0 &= \frac{\bar{v}_2 - \bar{v}_1}{jx_{12}} + \frac{\bar{v}_2 - \bar{v}_3}{jx_{23}} - \bar{i}_2 \\
0 &= \frac{\bar{v}_3 - \bar{v}_1}{jx_{13}} + \frac{\bar{v}_3 - \bar{v}_2}{jx_{23}} - \bar{i}_3
\end{aligned}
\tag{8}
$$

Example 1 (III)

- In vector form:

$$\begin{bmatrix} \bar{i}_1 \\ \bar{i}_2 \\ 0 \end{bmatrix} = \left(\bar{\boldsymbol{Y}} + \boldsymbol{I}_3 \begin{bmatrix} 0 \\ 0 \\ 1/\bar{z}_3 \end{bmatrix} \right) \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{bmatrix} = \bar{\boldsymbol{Y}}_{\text{tot}} \bar{\boldsymbol{v}} \tag{9}$$

Example 1 (IV)

- where $\boldsymbol{I}_3$ is a 3×3 identity matrix and $\bar{\boldsymbol{Y}}$ is the so-called admittance matrix:

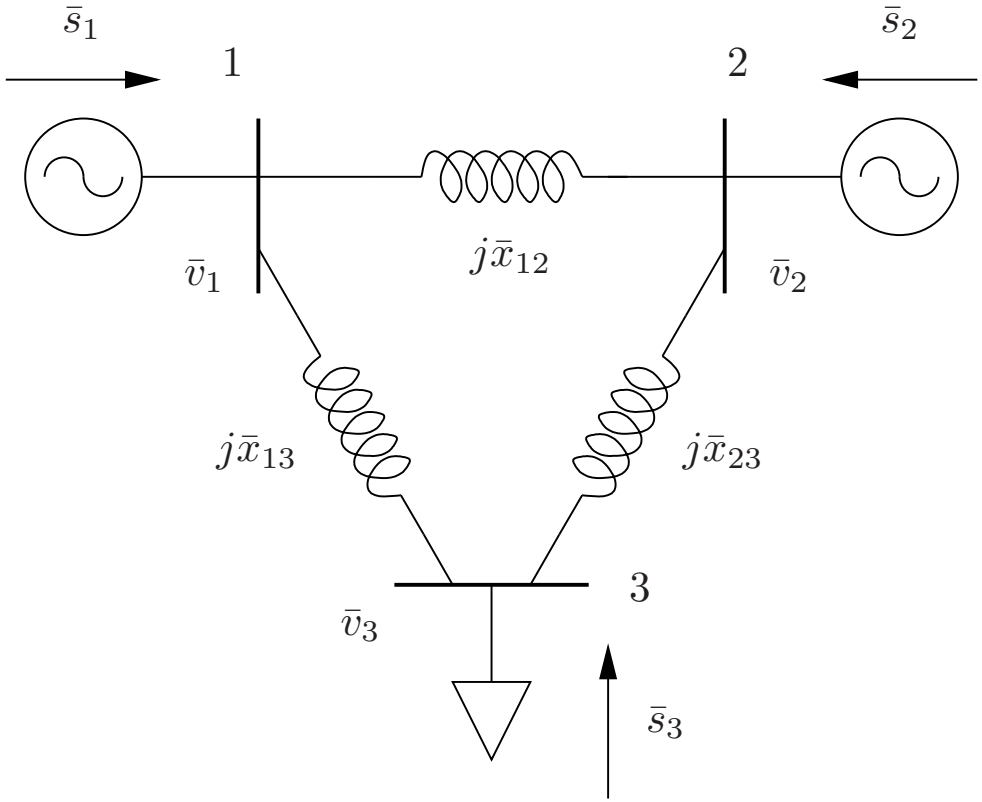
$$\bar{\boldsymbol{Y}} = \begin{bmatrix} 1/jx_{12} + 1/jx_{13} & -1/jx_{12} & -1/jx_{13} \\ -1/jx_{12} & 1/jx_{12} + 1/jx_{23} & -1/jx_{23} \\ -1/jx_{13} & -1/jx_{23} & 1/jx_{13} + 1/jx_{23} \end{bmatrix} \quad (10)$$

Background (II)

- The power flow problem is conceptually the same problem as solving a steady-state ac circuit.
- The only, though substantial, difference is the set of input data.
- Loads are expressed in terms of consumed active and reactive powers (*PQ load*) and generators are defined in terms of constant voltage magnitude and active power injection (*PV generator*).
- Hence, the power flow problem is **nonlinear**.

Example 2 (I)

- Classical power flow problem.



Example 2 (II)

- The power flow problem is formulated in order to determine unknown voltage magnitudes and angles.

$$
\begin{aligned}
0 &= \frac{v_2 v_1}{x_{12}} \sin(\theta_2 - \theta_1) - p_2 \\
0 &= \frac{v_3 v_1}{x_{13}} \sin(\theta_3 - \theta_1) + \frac{v_3 v_2}{x_{23}} \sin(\theta_3 - \theta_2) - p_3 \\
0 &= \frac{v_3^2}{x_{13}} + \frac{v_3^2}{x_{23}} - \frac{v_3 v_1}{x_{13}} \cos(\theta_3 - \theta_1) - \frac{v_3 v_2}{x_{23}} \cos(\theta_3 - \theta_2) - q_3
\end{aligned}
\tag{11}
$$

where the unknowns are v_3, θ_3 and θ_2.

Rationales Behind Power Flow Problem Formulation

- Loads at high voltage level are modelled as constant PQ due to under-load tap changers.
- Generators are modelled as PV due to turbine and voltage regulators.
- Transmission lines and transformers are generally modelled as lumped π-circuits with constant parameters.
- Observe that one bus has to be the phase angle reference. Typically, one generator is used as *slack* bus.
- The *distributed* slack bus model is more physical.

General Formulation of the Power Flow Problem (I)

- The vector of currents injected at each node is:

$$\bar{\boldsymbol{i}} = \bar{\boldsymbol{Y}}\bar{\boldsymbol{v}} \tag{12}$$

which leads to write the power flow problem as the complex power injections at buses:

$$\bar{\boldsymbol{s}} = \bar{\boldsymbol{V}}\bar{\boldsymbol{i}}^* = \bar{\boldsymbol{V}}\bar{\boldsymbol{Y}}^*\bar{\boldsymbol{v}}^* \tag{13}$$

where $\bar{\boldsymbol{V}} = \text{diag}(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{n_b})$ and n_b is the number of network buses.

General Formulation of the Power Flow Problem (II)

- Rewriting (13):

$$\mathbf{0} = \bar{\boldsymbol{s}} - \bar{\boldsymbol{V}}\bar{\boldsymbol{Y}}^{*}\bar{\boldsymbol{v}}^{*} \tag{14}$$

- Or, more in general:

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}) \tag{15}$$

where $\boldsymbol{y}$ and partameters per bus type are:

Bus type	Variables	Data
Slack generator	p, q	v, θ
PV generator	q, θ	p, v
PQ load	v, θ	p, q

Relevant Issues of the Power Flow Problem

- The origins of the formulation of the power flow problem and the solution based on the Newton's method date back to the late sixties.
- Since then, a huge variety of studies have been presented about the solution of the power flow problem, addressing:
  - Starting initial guess
  - Computational efficiency
  - Ill-conditioned cases
  - Robustness
  - Multiple solutions
  - Unsolvable cases

Taxonomy of the Power Flow Problem

- It is relevant to classify the power flow problems into the following categories:
  - Well-conditioned case
  - Ill-conditioned case
  - Bifurcation point
  - Non-physical solution
  - Unsolvable case

Well-conditioned case

- The power flow solution exists and is reachable using a flat initial guess (e.g., all load voltage magnitudes equal to 1 and all bus voltage angles equal to 0) and a standard Newton's method.
- This case is the most common situation.

Ill-conditioned case

- The solution of the power flow problem does exist, but standard solvers fail to get this solution starting from a flat initial guess.
- This situation is due to the fact that the region of attraction of the power flow solution is narrow or far from the initial guess.
- In this case, the failure of standard power flow procedure is due to the instability of the numerical method, not of the power flow equations.
- Robust power flow methods have proved to be efficacious for solving ill-conditioned cases.

Bifurcation Point

- The solution of the power flow exists but it is either a saddle-node bifurcation or a limit-induced bifurcation.
  - Saddle-node bifurcations are associated with the maximum loading condition of a system. The solution cannot be obtained using standard or robust power flow methods, since the power flow Jacobian matrix is singular at the solution point.
  - Limited-induced bifurcations are associated with a physical limit of the system, such as the shortage of generator reactive power. The solution point is typically a well-conditioned case and does not show convergence issues.

Non-physical Solution

- Solutions that cannot be accepted since some variable is out of its technical limits.
- Typically these solution are characterized by very low voltage levels.
- These solutions are also known as: *extraneous*, *false*, *lower*, or *unstable*.

Unsolvable Case

- The solution of the power flow problem does not exist.
- Typically, the issue is that the loading level of the network is too high.
- As in the case of the bifurcation points, a continuation method or an optimal power flow problem allows defining the maximum loading level that the system can supply.

Standard Power Flow Solution Methods

Solution Methods (I)

- Methods that do not require the computation of the Jacobian matrix of $\boldsymbol{g}$:
  - Jacobi's method.
  - Gauss-Seidel's method.

- Methods that require the computation of the Jacobian matrix of $\boldsymbol{g}$:
  - Newton's (or Newton-Raphson's) method.
  - Robust Newton's methods.

Solution Methods (II)

- Methods that simplify the Jacobian matrix of $\boldsymbol{g}$:
  - Inexact and dishonest Newton's methods.
  - Fast decoupled power flow.

- Methods that simplify $\boldsymbol{g}$:
  - DC power flow model.

Newton-Raphson's Method (I)

- The i-th iteration of the classical Newton's method for (15) is as follows:

$$\begin{aligned} \Delta \boldsymbol{y}^{(i)} &= -[\boldsymbol{g}_{\boldsymbol{y}}^{(i)}]^{-1} \boldsymbol{g}^{(i)} \\ \boldsymbol{y}^{(i+1)} &= \boldsymbol{y}^{(i)} + \Delta \boldsymbol{y}^{(i)} \end{aligned} \tag{16}$$

where $\boldsymbol{g}^{(i)} = \boldsymbol{g}(\boldsymbol{y}^{(i)})$, $\boldsymbol{g}_{\boldsymbol{y}}^{(i)} = \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{y}^{(i)})$, and $\boldsymbol{g}_{\boldsymbol{y}} = \nabla_{\boldsymbol{y}}^{T} \boldsymbol{g}$ is the Jacobian matrix of the power flow equations.

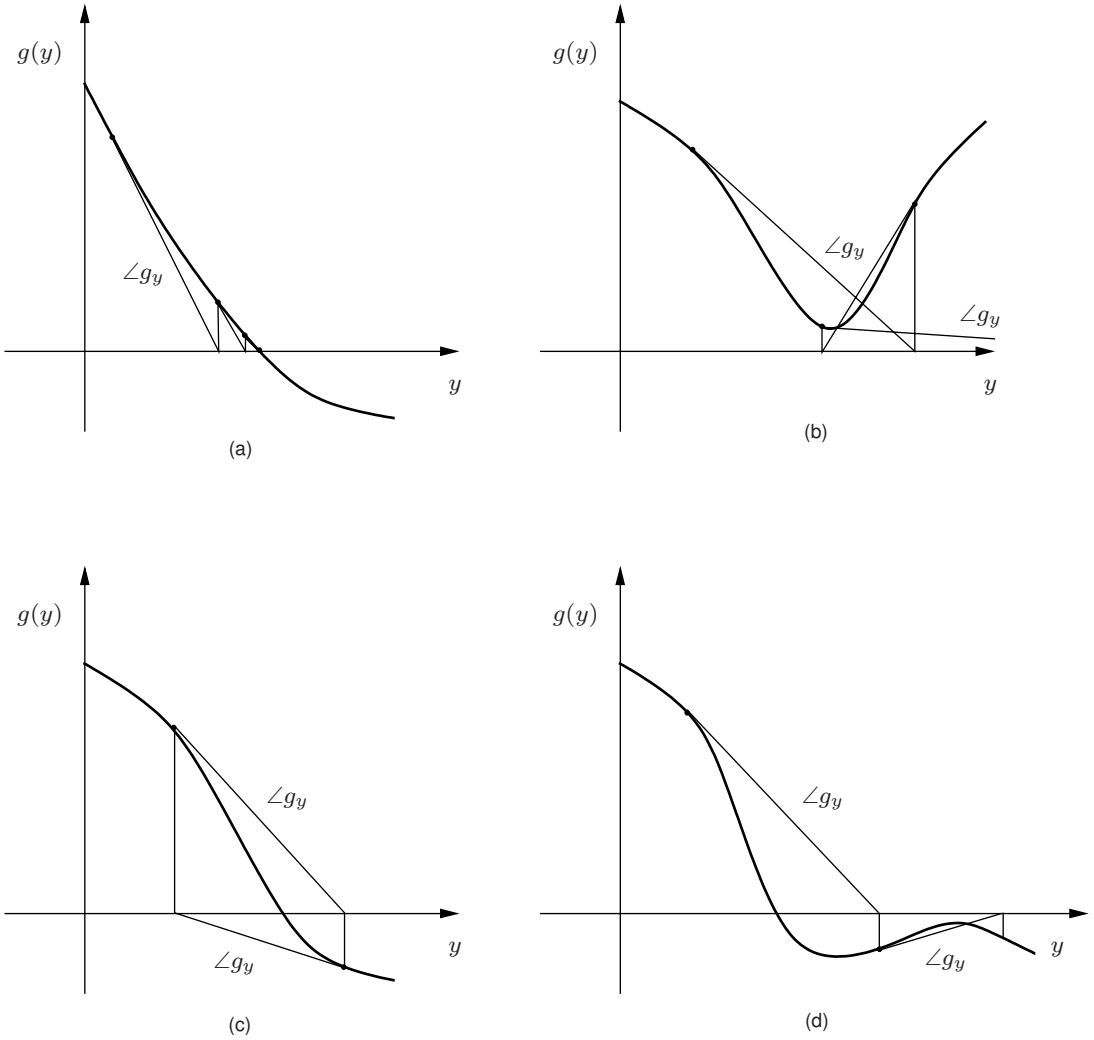
Newton-Raphson's Method (II)

- The geometrical interpretation of the Newton's method is well-known. For the actual value $\boldsymbol{y}^{(i)}$, one computes the tangent of $\boldsymbol{g}^{(i)}$ as:

$$\boldsymbol{\tau}(\boldsymbol{y}) = \boldsymbol{g}^{(i)} + \boldsymbol{g}_{\boldsymbol{y}}^{(i)}(\boldsymbol{y} - \boldsymbol{y}^{(i)}) \tag{17}$$

- Imposing $\boldsymbol{\tau}(\boldsymbol{y}) = \boldsymbol{0}$ yields the value $\boldsymbol{y}^{(i+1)}$ defined in (16).

Geometrical Interpretation of the Newton-Raphson's Method



Robust Newton's Method (I)

- There are idiosyncratic cases for which the Newton's technique fails to converge.
- A variety of *robust* variations of the basic Newton's method have been proposed in the literature for solving ill-conditioned cases.
- The majority of these techniques mainly consist in modifying the first equation of (16) as follows:

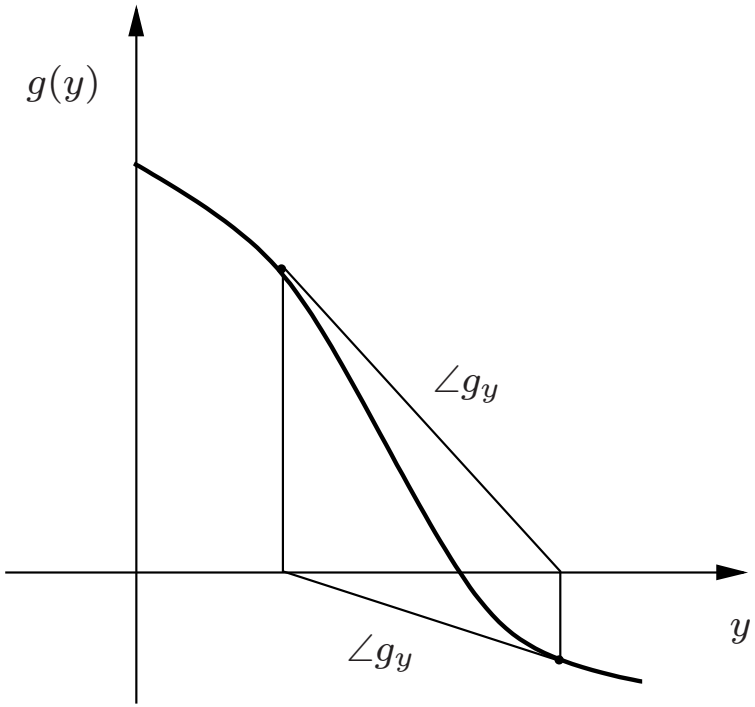
$$\Delta \boldsymbol{y}^{(i)} = -\alpha [\boldsymbol{g}_{\boldsymbol{y}}^{(i)}]^{-1} \boldsymbol{g}^{(i)} \tag{18}$$

where α is a factor that improves the convergence properties of the iterative process.

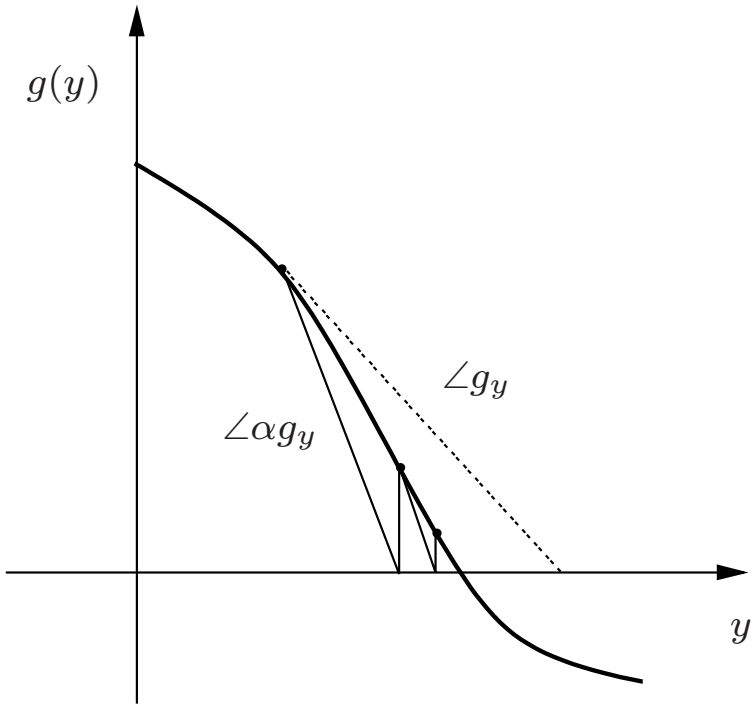
Robust Newton's Method (II)

- If α is the result of an optimization process, α is called *optimal multiplier* (e.g., Iwamoto's method).
- It is important not to confuse ill-conditioned cases with those that are unsolvable since the solution does not exist.
- Robust solvers are useful in case of ill-conditioned systems but do not generally work well for unsolvable cases.
- Unsolvable cases are better tackled using the continuation power flow technique.

Geometrical Interpretation of the Robust Newton's Method



(a)



(b)

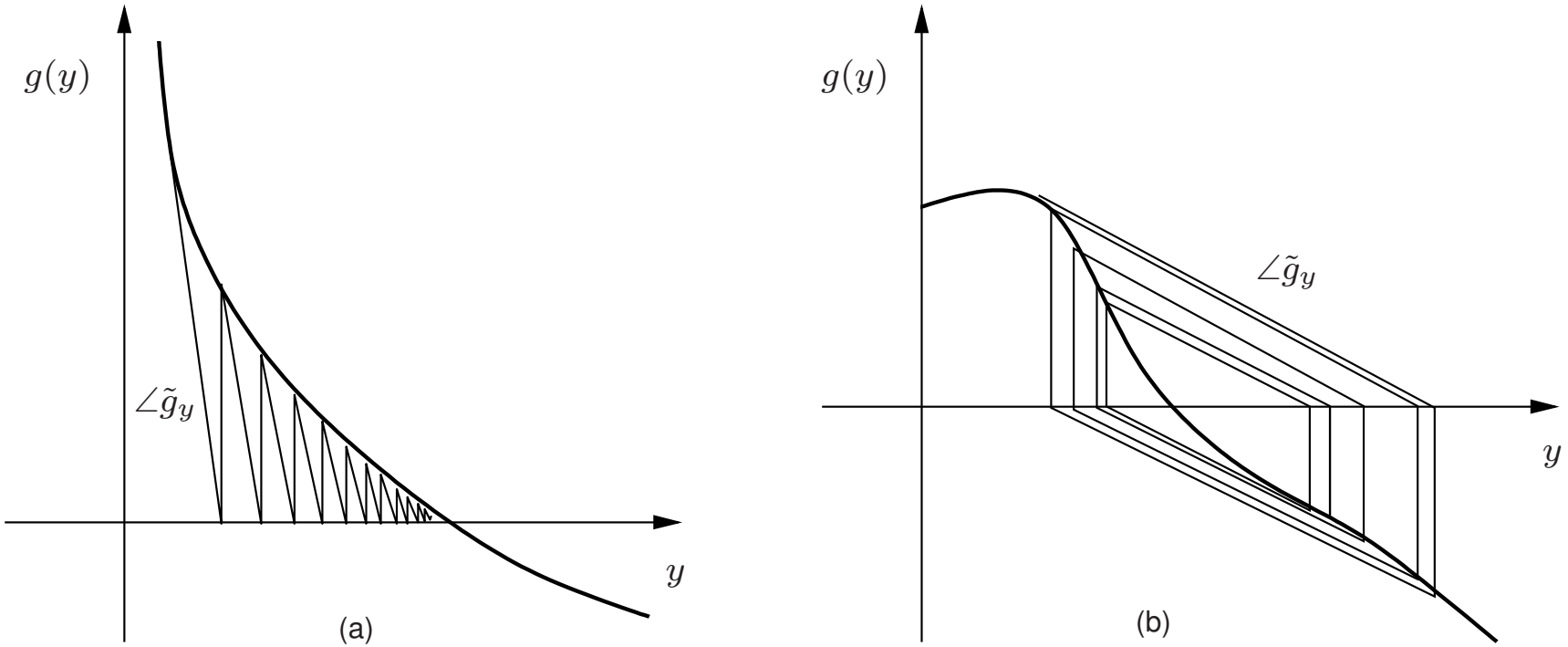
Main Issue of the Newton's Method

- One of the most relevant drawbacks of the Newton's method is the need of factorizing the full Jacobian matrix at each iteration.
- From the computational point of view, the factorization of a matrix is an order N^3 operation, i.e., the computational weight increases with the cube of the size N of the matrix.
- The computational effort can be reduced to $N^{1.5}$ if using sparse matrices techniques, which allows saving a considerable time for large systems (e.g., thousands of buses).
- However, the Jacobian matrix factorization remains the most critical issue of the Newton's method (about 85% of the total CPU time for networks with thousands of buses).

Inexact and Dishonest Newton's Method

- Inexact methods aim to approximate the factorization of the Jacobian matrix.
- A family of inexact methods are based on the Generalized Minimal Residual (GMRES) method.
- The GMRES is a particular case of Krylov's subspace. The main issue is to properly pre-conditioning the Jacobian matrix.
- Dishonest methods compute the Jacobian matrix factorization only the first one or two iterations and then use the previous factorization for the remaining iterations.

Geometrical Interpretation of the Dishonest Newton's Method



Fast Decoupled Power Flow (FDPF)

- The FDPF is a particular case of dishonest Newton's method.
- The Jacobian matrix is approximated so that it becomes block diagonal and all non-zero elements are constant. Hence only one factorization is needed.
- The FDPF requires much more iterations than the NR method but proved to be more robust.

Comparison of Power Flow Solution Methods (I)

	Newton		Jacobi		Gauss-Seidel	
Bus #	Iter. #	time [s]	Iter. #	time [s]	Iter. #	time [s]
14	4	0.0050	76	0.0217	56	0.0288
118	5	0.0287	580	0.505	388	2.738
1228	5	0.210	454	5.120	224	112.4
11856	4	3.15	340	399.0	173	9112

Comparison of Power Flow Solution Methods (II)

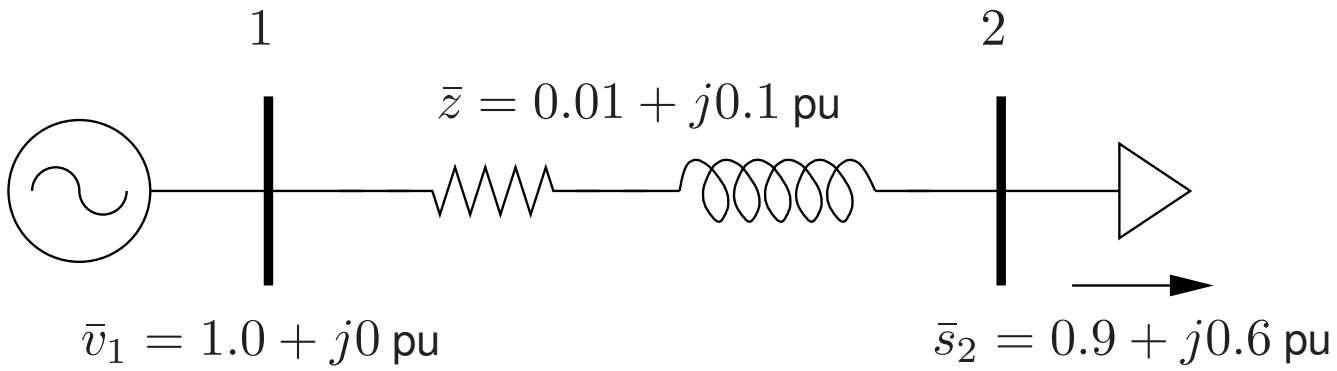
	GMRES		Dishonest		FDPF	
Bus #	Iter. #	time [s]	Iter. #	time [s]	Iter. #	time [s]
14	4	0.4339	7	0.0040	6	0.0053
118	7	53.53	15	0.0183	6	0.0117
1228	n.a.	n.a.	26	0.207	12	0.160
11856	n.a.	n.a.	10	3.820	5	5.174

Region of Attraction (I)

- A key issue of any iterative technique is the initial guess.
- The only way to know if a given initial guess is adequate for obtaining a solution $\boldsymbol{y}_0$ of the power flow problem is to determine the region of attraction of $\boldsymbol{y}_0$. At this regard, the initial guess can be of three types:
  - The initial guess is inside the region of attraction of the solution $\boldsymbol{y}_0$ and the numerical method converges.
  - The initial guess is outside the region of attraction of the solution $\boldsymbol{y}_0$. Numerical methods typically diverge if one starts with such initial guess.
  - Although the initial guess is within the region of attraction, the numerical method diverges.

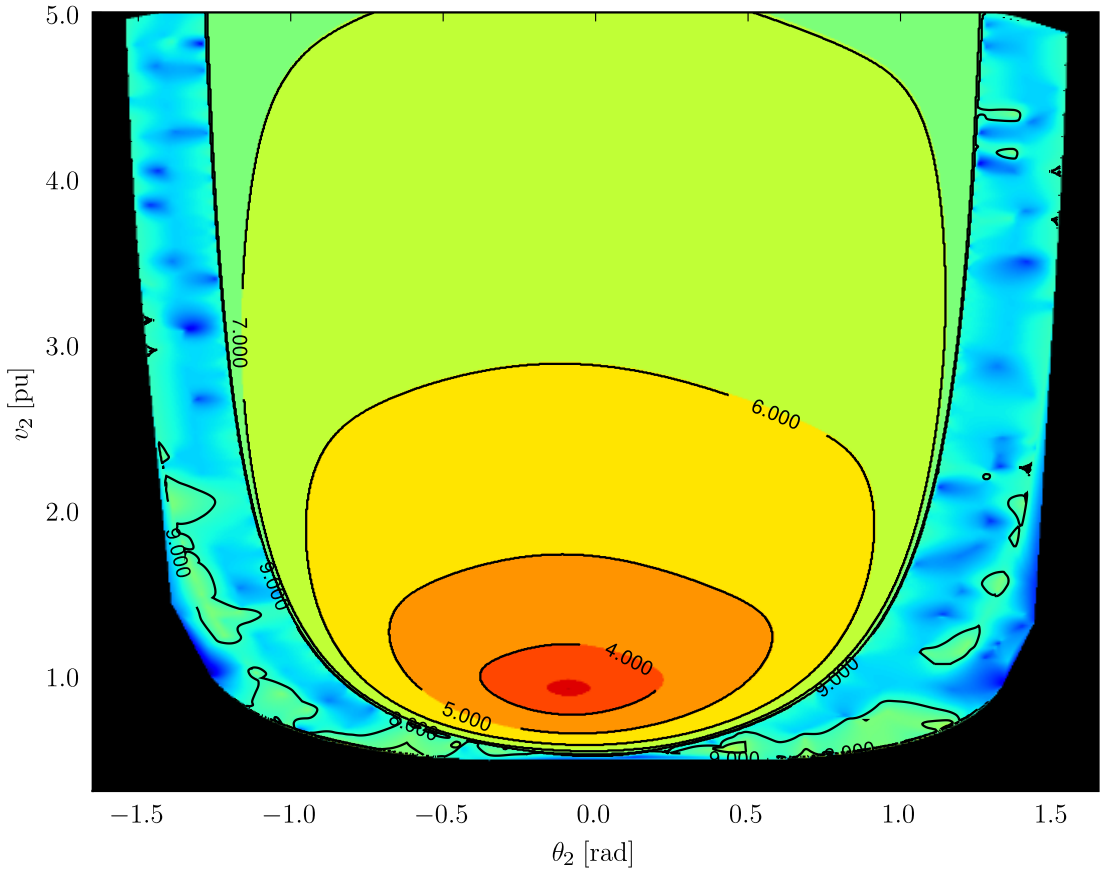
Example 3 (I)

- Consider the following simple 2-bus system.



Example 3 (II)

- Region of attraction of the Newton's method for a 2-bus system.



Region of Attraction (II)

- Different methods have different region of attractions.
- Unfortunately defining the region of attraction is extremely costly.
- In practice it is virtually impossible to define the region of attraction for a real-world system.
- *Robust* methods are thus required.

Continuous Newton's Method

Ordinary Differential Equations (ODE)

- Let us consider a set of autonomous ODE:

$$\dot{\boldsymbol{y}} = \tilde{\boldsymbol{f}}(\boldsymbol{y}) \tag{19}$$

The simplest method of numerical integrating (19) is the explicit Euler's method, as follows:

$$\begin{aligned} \Delta \boldsymbol{y}^{(i)} &= \Delta t \tilde{\boldsymbol{f}}(\boldsymbol{y}^{(i)}) \\ \boldsymbol{y}^{(i+1)} &= \boldsymbol{y}^{(i)} + \Delta \boldsymbol{y}^{(i)} \end{aligned} \tag{20}$$

where Δt is a given step length.

Newton-Raphson's Method (reprise)

- Let's recall the i-th iteration of the classical Newton's method:

$$\begin{aligned} \Delta \boldsymbol{y}^{(i)} &= -[\boldsymbol{g}_{\boldsymbol{y}}^{(i)}]^{-1} \boldsymbol{g}^{(i)} \\ \boldsymbol{y}^{(i+1)} &= \boldsymbol{y}^{(i)} + \Delta \boldsymbol{y}^{(i)} \end{aligned} \tag{21}$$

where $\boldsymbol{g}^{(i)} = \boldsymbol{g}(\boldsymbol{y}^{(i)})$, $\boldsymbol{g}_{\boldsymbol{y}}^{(i)} = \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{y}^{(i)})$, and $\boldsymbol{g}_{\boldsymbol{y}} = \nabla_{\boldsymbol{y}}^{T} \boldsymbol{g}$ is the Jacobian matrix of the power flow equations.

Analogy between Euler's and Newton's Methods

- The analogy between (21) and (20) is straightforward if one defines:

$$\tilde{\boldsymbol{f}}(\boldsymbol{y}) = -[\boldsymbol{g}_{\boldsymbol{y}}]^{-1}\boldsymbol{g}(\boldsymbol{y}) \tag{22}$$

  Equations (21) can thus be viewed as the i^{th} step of the Euler's forward method where $\Delta t = 1$.

- Furthermore, robust Newton's methods are nothing but the i^{th} step of the Euler's integration method where $\Delta t = \alpha$.

- In other words, the computation of the optimal multiplier α corresponds to a variable step forward Euler's method.

Continuous Newton's Method

- Equations (19) and (22) leads to:

$$\dot{\boldsymbol{y}} = -[\boldsymbol{g_y}]^{-1}\boldsymbol{g}(\boldsymbol{y}) \tag{23}$$

  which is known as *continuous Newton's method*.

- The equilibrium point $\boldsymbol{y}_0$ of (23) is

$$\mathbf{0} = \tilde{\boldsymbol{f}}(\boldsymbol{y}_0) = -[\boldsymbol{g_y}|_0]^{-1}\boldsymbol{g}(\boldsymbol{y}_0) \tag{24}$$

  Thus, assuming that $\boldsymbol{g_y}$ is not singular, $\boldsymbol{y}_0$ is also the solution of the power flow problem.

Stability of the Continuous Newton's Method (I)

- Differentiating (22) with respect to $\boldsymbol{y}$ leads to:

$$\begin{aligned} \tilde{\boldsymbol{f}}_{\boldsymbol{y}} &= \nabla_{\boldsymbol{y}}^{T} \tilde{\boldsymbol{f}}(\boldsymbol{y}) \\ &= -[\boldsymbol{g}_{\boldsymbol{y}}]^{-1} \boldsymbol{g}_{\boldsymbol{y}} - (\nabla_{\boldsymbol{y}}^{T}([\boldsymbol{g}_{\boldsymbol{y}}]^{-1}))\boldsymbol{g}(\boldsymbol{y}) \\ &= -\boldsymbol{I}_{n_y} - (\nabla_{\boldsymbol{y}}^{T}([\boldsymbol{g}_{\boldsymbol{y}}]^{-1}))\boldsymbol{g}(\boldsymbol{y}) \end{aligned} \tag{25}$$

  where $\boldsymbol{I}_{n_y}$ is the identity matrix of order n_y. Since the equilibrium point $\boldsymbol{y}_0$ is a solution for $\boldsymbol{g}(\boldsymbol{y}_0) = 0$, one has:

$$\tilde{\boldsymbol{f}}_{\boldsymbol{y}}|_0 = -\boldsymbol{I}_{n_y} \tag{26}$$

Stability of the Continuous Newton's Method (II)

- Equation (26) implies that all eigenvalues of $\tilde{\boldsymbol{f}}_{\boldsymbol{y}}$ at the solution point are equal to -1.
- (26) means that the solution of (23), if exists, is asymptotically stable.
- The reachability of this solution depends on the starting point $\boldsymbol{y}(t_0) = \boldsymbol{y}^{(0)}$, which has to be within the *region of attraction* or *stability region* of the equilibrium point $\boldsymbol{y}_0$.
- The continuous Newton's method is expected to show better ability to converge than other methods previously discussed if the initial guess is within the region of attraction.

Example 5

- The previous result is straightforward for a scalar $g(y)$, i.e. for $y \in \mathbb{R}$ and $g \in \mathbb{R}$, as follows:

$$\dot{y} = \tilde{f}(y) = -\frac{g(y)}{g_y(y)} \tag{27}$$

$$\Rightarrow \quad \tilde{f}_y(y) = -\frac{g_y(y)}{g_y(y)} + \frac{g_{yy}(y)}{g_y^2(y)} g(y) \tag{28}$$

$$= -1 + \frac{g_{yy}(y)}{g_y^2(y)} g(y)$$

thus $\tilde{f}_y(y_0) = -1$ if $g(y_0) = 0$ and $g_y(y_0) \neq 0$.

A General Framework for Power Flow Solvers

- It is well-known that the forward Euler's method, even with variable time step, can be numerically unstable.
- Given the analogy between the power flow equations and an ODE system, any well-assessed numerical method can be used for integrating (23).
- It is thus intriguing to use some efficient integration method for solving (23).

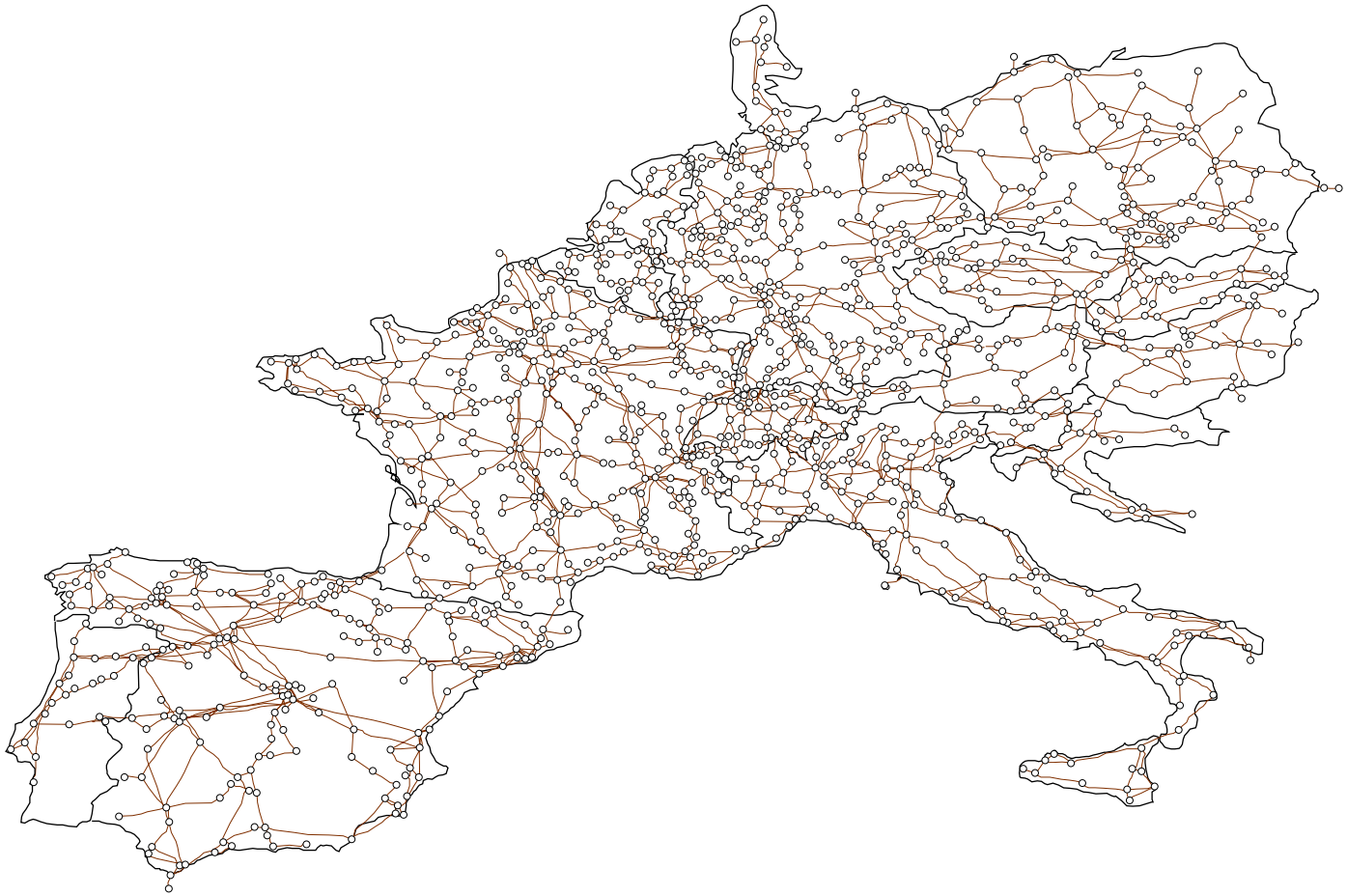
Runge-Kutta Formulas

- Since computing the Jacobian matrix of (23) is complex, explicit ODE integration methods should be preferred.
- Runge-Kutta formulas are good candidates.
- For example the classic RK4 formula is as follows:

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(\boldsymbol{x}^{(i)}) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(\boldsymbol{x}^{(i)} + 0.5\Delta t \boldsymbol{k}_1) \\ \boldsymbol{k}_3 &= \boldsymbol{f}(\boldsymbol{x}^{(i)} + 0.5\Delta t \boldsymbol{k}_2) \\ \boldsymbol{k}_4 &= \boldsymbol{f}(\boldsymbol{x}^{(i)} + \Delta t \boldsymbol{k}_3) \\ \boldsymbol{x}^{(i+1)} &= \boldsymbol{x}^{(i)} + \Delta t(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)/6 \end{aligned} \tag{29}$$

Example 6 (I)

- Let consider the 1254-bus model of the UCTE system.



Example 6 (II)

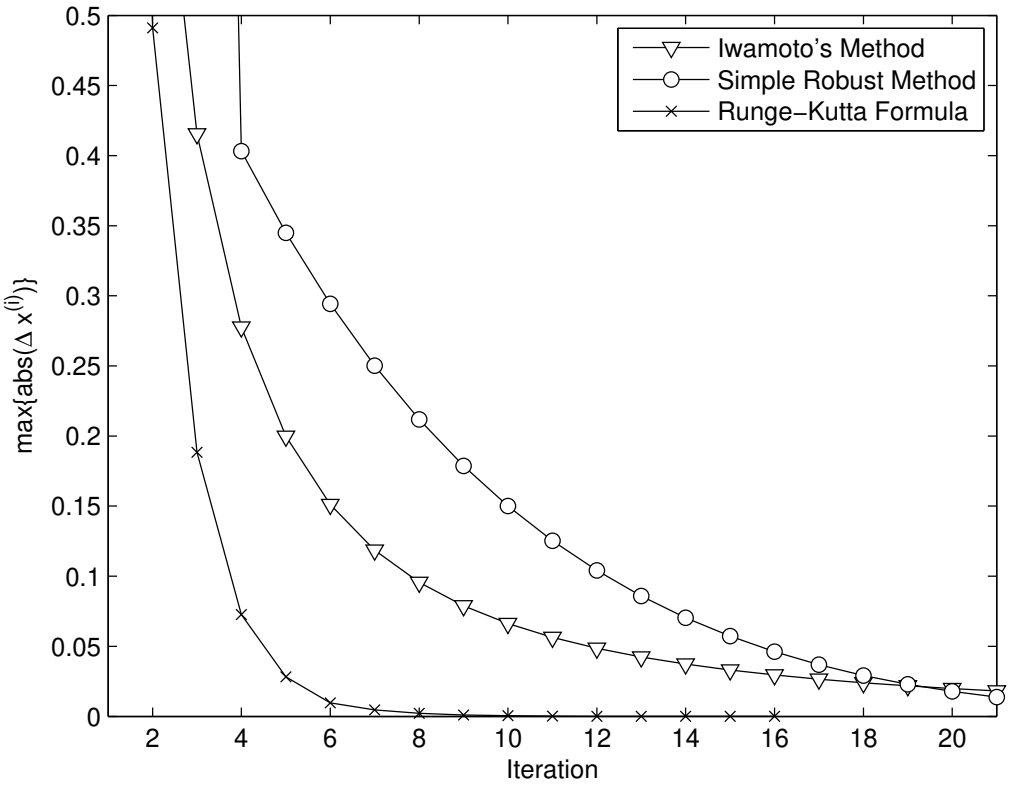
- Comparison of methods for solving the power flow of the UCTE system.

Method	# Iter. $\epsilon = 10^{-3}$	# Iter. $\epsilon = 10^{-4}$	# Iter. $\epsilon = 10^{-5}$
Standard NR	-	-	-
Fast Decoupled PF	-	-	-
Iwamoto's method	99	320	1021
Simple robust method	31	39	47
Runge-Kutta method	10	13	16

where ϵ is the required convergence error tolerance.

Example 6 (III)

- Comparison of convergence errors obtained with different robust power flow solution methods for the UCTE system.



Example 6 (IV)

- Comparison of the computational burden of different solution techniques for the UCTE system.

Method	CPU time (s)
Iwamoto's method	106.5
Simple robust method	3.5
Runge-Kutta method	3.4

References

- Further insights and references to all topics covered in the talk can be found in:

F. Milano, Continuous Newton's Method for Power Flow Analysis, IEEE Transactions on Power Systems, Vol. 24, No. 1, pp. 50-57, February 2009.

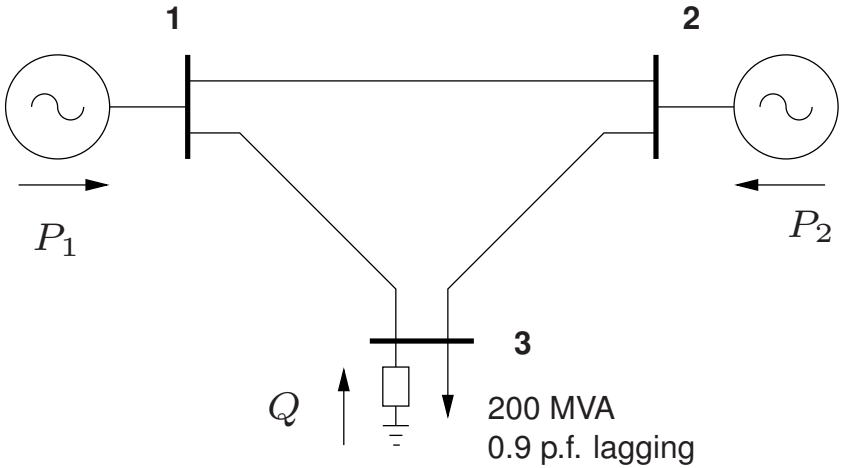
Available at:

faraday1.ucd.ie/publications.html

Example

Example

- For the following system:



All lines:

200 MVA

138 kV

$X = 0.1$ p.u.

$B = 0.2$ p.u.

$V_1 = 1, \delta_1 = 0, V_2 = 1, V_3 = 1$, and $P_2 = P_1/2$.

- Determine the voltage phasor angles δ_2 and δ_3 and the shunt Q by solving the PF equations.

Example

- The $\mathbf{Y}_{bus}$ matrix is:

$$\begin{aligned}
\mathbf{Y}_{11} &= \mathbf{Y}_{22} = \mathbf{Y}_{33} = \frac{2}{jX} + 2\left(j\frac{B}{2}\right) = -j19.8 \\
\mathbf{Y}_{12} &= \mathbf{Y}_{13} = \mathbf{Y}_{23} = -\frac{1}{jX} = j10
\end{aligned}$$

$$\Rightarrow \quad \mathbf{Y}_{bus} = j\begin{bmatrix} -19.8 & 10 & 10 \\ 10 & -19.8 & 10 \\ 10 & 10 & -19.8 \end{bmatrix} = j\begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

Example

- Mismatch equation ΔP_1:

$$
\begin{aligned}
\Delta P_1 &= P_1 - \sum_{k=1}^{3} V_1 V_k [G_{1k} \cos(\delta_1 - \delta_k) + B_{1k} \sin(\delta_1 - \delta_k)] \\
0 &= P_1 - \sum_{k=1}^{3} B_{1k} \sin(-\delta_k) \\
&= P_1 + B_{12} \sin(\delta_2) + B_{13} \sin(\delta_3) \\
&= P_1 + 10 \sin(\delta_2) + 10 \sin(\delta_3)
\end{aligned}
$$

Example

- Mismatch equation ΔQ_1:

$$\begin{aligned}
\Delta Q_1 &= Q_1 - \sum_{k=1}^{3} V_1 V_k [G_{1k} \sin(\delta_1 - \delta_k) - B_{1k} \cos(\delta_1 - \delta_k)] \\
0 &= Q_1 + \sum_{k=1}^{3} B_{1k} \cos(-\delta_k) \\
&= Q_1 - 19.8 + 10 \cos(\delta_2) + 10 \cos(\delta_3)
\end{aligned}$$

Example

- Mismatch equation ΔP_2:

$$\begin{aligned}
\Delta P_2 &= P_2 - \sum_{k=1}^{3} V_2 V_k [G_{2k} \cos(\delta_2 - \delta_k) + B_{2k} \sin(\delta_2 - \delta_k)] \\
0 &= P_1/2 - \sum_{k=1}^{3} B_{2k} \sin(\delta_2 - \delta_k) \\
&= 0.5 P_1 + 10 \sin(\delta_2) + 10 \sin(\delta_2 - \delta_3)
\end{aligned}$$

Example

- Mismatch equation ΔQ_2:

$$\begin{aligned}
\Delta Q_2 &= Q_2 - \sum_{k=1}^{3} V_2 V_k [G_{2k} \sin(\delta_2 - \delta_k) - B_{2k} \cos(\delta_2 - \delta_k)] \\
0 &= Q_2 + \sum_{k=1}^{3} B_{2k} \cos(\delta_2 - \delta_k) \\
&= Q_1 + 10 \cos(\delta_2) - 19.8 + 10 \cos(\delta_2 - \delta_3)
\end{aligned}$$

Example

- Mismatch equation ΔP_3:

$$\begin{aligned}
\Delta P_3 &= P_3 - \sum_{k=1}^{3} V_3 V_k [G_{3k} \cos(\delta_3 - \delta_k) + B_{3k} \sin(\delta_3 - \delta_k)] \\
0 &= -0.9 - \sum_{k=1}^{3} B_{3k} \sin(\delta_3 - \delta_k) \\
&= -0.9 + 10 \sin(\delta_3) + 10 \sin(\delta_3 - \delta_2)
\end{aligned}$$

Example

- Mismatch equation ΔQ_3:

$$\begin{aligned}
\Delta Q_3 &= Q_3 - \sum_{k=1}^{3} V_3 V_k [G_{3k} \sin(\delta_3 - \delta_k) - B_{3k} \cos(\delta_3 - \delta_k)] \\
0 &= Q - \sqrt{1 - 0.9^2} + \sum_{k=1}^{3} B_{3k} \cos(\delta_3 - \delta_k) \\
&= Q - 0.436 + 10 \cos(\delta_3) + 10 \cos(\delta_3 - \delta_2) - 19.8
\end{aligned}$$

Example

- Thus 6 equations and 6 unknowns, i.e. δ_2, δ_3, P_1, Q_1, Q_2, and Q, can be solved using MATLAB's `fsolve()` routine:

```
>> global lambda
>> lambda = 1;
>> z0 = fsolve(@pf_eqs,[0 0 0 0 0 0], optimset('Display','iter'))

                                         Norm of    First-order  Trust-region
 Iteration  Func-count     f(x)          step       optimality   radius
     0          7        0.945696                        18             1
     1         14     0.000661419     0.705901        0.0205             1
     2         21     9.98637e-18    0.0257331      2.52e-09          1.76
Optimization terminated: first-order optimality is less than options.TolFun.

z0 =

   -0.0100   -0.0500    0.6000   -0.1870   -0.1915    0.2565
```

Example

- Where `lambda` is used to simulate constant power factor load changes and `pf_eqs.m` is:

```
function F = pf_eqs(z)

global lambda

d2 = z(1);
d3 = z(2);
P1 = z(3);
Q1 = z(4);
Q2 = z(5);
Q  = z(6);

F(1,1) = P1 + 10*sin(d2) + 10*sin(d3);
F(2,1) = Q1 - 19.8 + 10*cos(d2) + 10*cos(d3);
F(3,1) = 0.5*P1 - 10*sin(d2) - 10*sin(d2-d3);
F(4,1) = Q2 + 10*cos(d2) - 19.8 + 10*cos(d2-d3);
F(5,1) = -0.9*lambda - 10*sin(d3) - 10*sin(d3-d2);
F(6,1) = Q - 0.436*lambda + 10*cos(d3) + 10*cos(d3-d2) - 19.8;
```

Example

- Observe that as the load (`lambda`) is increased, convergence problems show up:

```
lambda = 20;
z0 = fsolve(@pf_eqs,[0 0 0 0 0 0], optimset('Display','iter'))

                                         Norm of    First-order  Trust-region
 Iteration  Func-count     f(x)          step       optimality   radius
     0          7          396.67                        360             1
     1         14          246.008            1          143             1
.
.
.
     9         70      2.68094e-07      0.179729     0.000311          25.9
    10         77      7.43614e-16    0.00127159     1.64e-08          25.9
Optimization terminated: first-order optimality is less than options.TolFun.

z0 =

   -0.2501   -1.2613   12.0000    7.0658    4.8031   20.1667
```

Example

- Convergence problems develop until the equations fail to converge:

```
lambda = 22;
z0 = fsolve(@pf_eqs,[0 0 0 0 0 0], optimset('Display','iter'))

                                         Norm of    First-order  Trust-region
 Iteration  Func-count     f(x)          step       optimality   radius
     0          7          480.33                        396             1
     1         14          310.93             1          168             1
.
.
.
    90        595       0.00622095     0.0211144       0.0322        0.0211
    91        602       0.00621755     0.0211144       0.0155        0.0211
Maximum number of function evaluations reached: increase options.MaxFunEvals.

z0 =

   -0.3322   -1.7225   13.1600   11.8633    8.5479   29.1072
```