CHALMERS
UNIVERSITY OF TECHNOLOGY

Investigation of efficient and reliable numerical algorithms for coupled reactor calculations

X-TREAM project: Task 1b
Survey of the state-of-the-art numerical techniques for solving coupled non-linear multi-physics equations.

Author:
Dr. Manuel Calleja
calleja@chalmers.se

Chalmers University of Technology
Department of Applied Physics
Division of Nuclear Engineering SE-4196 Göteborg

CTH-NT-289 August 14, 2014

Abstract

Nowadays, the precise modeling of a nuclear reactor core is a challenge. This task involves several aspects, from the computational power needed to perform simulations, to the physics and analysis of the outcome. The need to better understand the physical phenomena is critical in order to quantify and qualify nuclear safety parameters. Currently, substantial research has been done in order to optimize the prediction capabilities of coupled codes. The need to better understand the multi-physics coupling between the neutronics and the thermal-hydraulics is required. In this report, state-of-the art of current methods and numerical techniques used in coupled codes are highlighted. A better understanding of the numerical schemes will allows selecting an appropriate algorithm to be implemented in POLCA-T. POLCA-T is the coupled code developed by Westinghouse that currently uses an explicit approach to couple the neutronics (POLCA7) and thermal-hydraulics (RIGEL). The final objective is to implement the concept of the Jacobian-free Newton-Krylov method, which will be used for solving the nonlinear equations which rise from the coupled solution.

Keywords: coupled multi-physics, coupling numerical schemes, JFNK method, neutronic and thermal-hydraulic interactions.

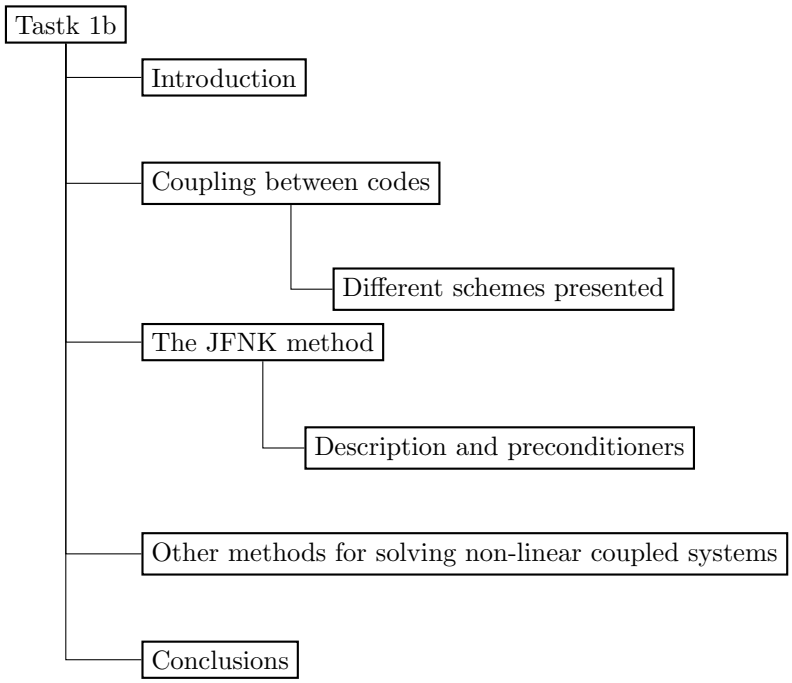


Figure 1: Structure of the report - Task 1b: Survey of the state of the art numerical techniques for solving coupled non-linear equations.

Contents

List of Figures

1 Acronyms

ABN:	Approximated Block Newton
ACMFDM:	Analytic Coarse Mesh finite Difference Method
ADF:	Assembly Discontinuity Factors
ATWC:	Anticipated Transient Without all Control rods
ANM:	Analytical Nodal Method
ATWS:	Anticipated Transient Without SCRAM
BiCGSTAB:	Bi-conjugated gradient
BWR:	Boiling Water Reactor
CFD:	Computational Fluid Dynamics
CMFD:	Coarse Mesh Finite Difference
DNBR:	Departure from Nuclear Boiling Ratio
FD:	Finite Differences
FV:	Finite Volumes
FEM:	Finite Elements Method
FDS:	Finite Difference Schemes
FDM:	Finite Differences Method
FEBE:	Forward Euler Backward Euler
FPI:	Fixed Point Iteration
GE:	General Electric
GMRES:	Generalized Minimal RESidual
HEM:	Homogeneous Equilibrium Model
HPLWR:	High Performance Light Water Reactor
HRM:	Homogeneous Relaxation Model
ICE:	Implicit Continuous Eulerian
INL:	Idaho National Laboratory
JFNK:	Jacobian-Free-Newton-Krylov
LOCA:	Loss Of Coolant Accident
LWR:	Light Water Reactor
MED:	Memory Exchage Data
MOX:	Mixed OXide
MSL:	Main Steam Line
MSLB:	Main Steam Line Break
N:	Neutronics
NEM:	Nodal Expansion Method
NPP:	Nuclear Power Plant
OS:	Operator Splitting
OSSI:	Operator Split Semi Implicit
PBP:	Physics Base Preconditioner
PDE:	Partial Differential Equation
PVM:	Parallel Virtual Machine
PWR:	Pressurized Water Reactor
RPV:	Reactor Pressure Vessel
SCRAM:	Safety Control Rod Axe Man
SLB:	Steam Line Break
SOR:	Successive Over Relaxation
TH:	Thermal-Hydraulics

2 Introduction

Accurate simulations for nuclear power plants (NPPs) have become more practical with the rapid growth of computing systems and improved algorithms. Especially, coupling of codes is a beneficial approach for safety analysis. The understanding of the interaction between neutronics (N) and thermal-hydraulics (TH) is important for achieving several objectives, e.g. the ability to perform safety analysis and the precise prediction of local parameters. In addition, numerical tools describing different physical phenomena have to be coupled with each other for analyzing e.g. normal plant conditions as well as, reactivity transients, and anticipated transients without SCRAM (ATWS).

The deterministic modeling performed today relies on the coupling of existing codes. The most mature multi-physical coupling scheme is the coupling of nodal diffusion codes with system codes and/or with 1D simplified TH codes using a rigid mapping scheme between codes. This coarse spatial discretization of both N and TH predicts unrealistic values of local phenomena. In fact, due to the multi-scale and multi-physics features of LWRs, detailed modeling of nuclear reactors for the whole system and all scales is still not feasible. The necessity to better describe the most important physical phenomena prevailing in modern core loadings of LWRs with increasing heterogeneity is driving the extension of current core design and N/TH coupled transient analysis methodologies. These improvements consider a more detailed spatial description of both N and TH computational domains and also the development of flexible coupling approaches as well as the use of improved diffusion or higher order approximations of the neutron transport equation.

In Sweden, research groups at KTH, Chalmers, and the industry with Westinghouse Electric Sweden AB, also lead the development of improved coupling approaches. The classical type of coupling approaches involves operator splitting methods. These methods solve each physics separately, passing data back and forth to achieve what is referred to as loose coupling. Unfortunately this approach has drawbacks, particularly when the coupled physics operate on disparate timescales or are very strongly coupled. An alternative for solving non-linear systems in a loosely coupled manner is to solve them simultaneously employing a tight coupled solution procedure. This procedure involves solving for all solution variables at the same time, usually through generation of one large nonlinear algebraic system that is solved using Newton's method. The type of research that the Division of Nuclear Engineering at Chalmers is pursuing, is based on the multi-physics nature of nuclear reactors by taking it into account from the beginning of the modeling process. Is worth it to mention, The Division of Nuclear Engineering at Chalmers has long been active in the deterministic modeling of nuclear reactors and a Deterministic REActor Modeling (DREAM) task force has being created. The present project is devoted to the investigation, improvement and development of numerical methods and techniques which take the multi-scale and multi-physics aspects of nuclear reactors into account. The benefit of this project would directly improve the economics, both from an operational point of view as well as from a reactor analysis point of view. The operational advantages would be the possibility to run the nuclear reactors closer to their safety limits, as a result of a better evaluation of the safety margins. This in itself would lead to a more sustainable use of the nuclear fuel and the corresponding natural resources. The main goal of the project is the development of innovative computing and modeling techniques for advanced nuclear reactor safety evaluations. Investigating different simulation strategies is targeted, which could later on be used in future simulation platforms. However, the sub-project that wraps and leads the current work is the X-TREAM project. This project, which stands for the "neXt generation numerical Techniques for deterministic REActor Modeling" is supported by NORTHNET (Nordic Thermal-Hydraulic Network), and will be carried out between 2013 and 2015.

The purpose of this report is to give a solid description of the current methodologies used for solving the coupled physics (N+TH). It is necessary to mention, that examples about these coupling schemes and other type of discretizations are given in a companion report named, "Investigation of efficient and reliable numerical algorithms for coupled reactor calculations," devoted to the task 1a of the previously mentioned X-TREAM project. The motivation of this work is based on the strong need to improve N and TH coupled systems, and to understand the techniques and physics behind this approach. The report is divided in sections, the first one emphasizing the common coupling techniques used nowadays. Section two describes the Jacobian-Free Newton Krylov (JFNK) method in addition to preconditioners. Following, other techniques for solving the system of non-linear equations are mentioned. Finally, the conclusion part makes an attempt to wrap up the work done and define the challenges in coupled calculations.

3 Survey of the state of the art numerical techniques for solving coupled non-linear equations

For both fuel assembly and pin-wise based coupling, the managing and proper adaptation of the time steps between the TH and N codes is important since both codes have their own time step selection algorithm. The coupling of two different codes has to be implemented for both stationary and transient conditions. The most widely used option is the loose ("low") coupled methods, which differently from tight ("high") [1], provide certain independence between the coupled codes. Here, different kinds of solutions are encountered, staggered operator splitting (OS) coupling (explicit type and also referred as "marching solution method" or "asynchronous models" [2]), semi-implicit [3], implicit schemes (also known as "synchronous models"), fixed point iteration (FPI) and the JFNK method. The nonlinearities between the N and TH models raises a challenge for finding an accurate and efficient solution for LWR transients. This tight coupling can be seen in the following equations.

$$\frac{1}{v_g}\frac{\partial \phi_g(r,t)}{\partial t} = D_g \frac{\partial^2 \phi_g(r,t)}{\partial r^2} - \Sigma_{T,g}(T)\phi_g(r,t) - \sum_{g'>g}^{G} \Sigma_{s,g'->g}(T)\phi_{g'}(r,t) + \sum_{g'=1}^{G} \chi_g \nu \Sigma_{f,g'}(T)(1-\beta)\phi_{g'}(r,t) + \sum_{i=1}^{I} \chi_{g,i}\lambda_i C_i(r,t). \tag{1}$$

$$\frac{\partial C_i(r,t)}{\partial t} = \beta_i \sum_{g=1}^{G} \nu \Sigma_{f,g}\phi_g(r,t) - \lambda_i C_i(r,t), i = 1,2,...,I \tag{2}$$

The N dependence on the TH is represented in the cross sections (fission, scattering and absorption), while the TH dependence on the N is foreseen via the heat source given by fission rates in the right hand side of equation 5 and 6, which affects directly the heat transfer equation applied to the heat structures.

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V}) = 0, \tag{3}$$

$$\frac{\partial \rho \vec{V}}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V} \otimes \vec{V}) = \vec{\nabla}\bar{\bar{\tau}} - \vec{\nabla} P + \rho \vec{g}, \tag{4}$$

$$\frac{\partial \rho U}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V} U) = -\vec{\nabla} q'', \tag{5}$$

$$\rho C_p \frac{\partial T}{\partial t} - \vec{\nabla} \cdot (k_f(T)\vec{\nabla} T) = q'''. \tag{6}$$

About these non-linearities for instance, the temporal and spatial discretizations affect stability parameters such as decay ratios and key parameters of reactivity insertion accidents (powers peak, Doppler temperatures and coolant enthalpies), which are very sensitive to the degree of implicitness in the coupling between the N and the TH models. Therefore, the different approaches for the temporal coupling between codes must be precisely understood. It is important to note that the spatial and temporal discretizations that each code follows is independent of the coupling approach between them, at least for those coupling approaches that do not solve the system of equations in a single block. In addition, some of the coupling issues involve:

1. Coupling approach - integration algorithm or parallel processing.
2. Modeling of the coupling - internal, external or parallel.
3. Spatial mesh overlays.
4. Coupling numerics - explicit, implicit, ... etc...
5. Coupled convergence schemes.

In the following sections, a series of numerical schemes used in the nuclear engineering field are presented. The focus however, is set to the JFNK methodology, due to its outstanding properties that facilitates the rapid development of multi-physics applications as a method for solving large scale non-linear systems.

3.1 Coupling between codes

For years, several N and TH codes have been coupled in order to understand the interaction between the two physics at different scales and following different schemes. In the following sections the common coupling algorithms are shown. It is worth to describe the parameters shown in figures 2, 3 and 4.

- N(NS) : Neutronic solution.
- TH(THS) : Thermal-Hydraulics solution.
- t_n : Current time step ("t_{n-1}" represents the previous time step).
- D_{mod} : Moderator density.
- T_{mod} : Moderator Temperature.
- T_{Dopp} : Fuel Doppler temperature.
- Pow : Total power.
- Δt : Time interval.

Moreover, reference to the previous report for task 1a of the X-TREAM project is sometimes needed, since precise examples of stand alone and coupled codes are given in that report.

3.1.1 Modeling techniques (applied to coupled systems)

While doing multi-physics calculations with N and TH codes, two critical parameters are encountered, the spatial mapping between the two domains (needed for the feedback exchange) and the type of interaction that both physics will achieve in order to obtain physical solutions. The three common types of coupling approaches are described in the next subsections[1].

Internal coupling: The objective of this coupling technique is to embed the codes and model the reactor core by both the N and TH systems. A system code is in charge of providing the boundary condition at the lower and upper plenum, the same which models the TH characteristics of the core, and a second code is in charge of modeling the N characteristics of the core.

External coupling: The boundary conditions are given by the system code and the core is fully modeled by a second code. This second code should be able to model the N(core model) and TH(core model) independently from the coolant system.

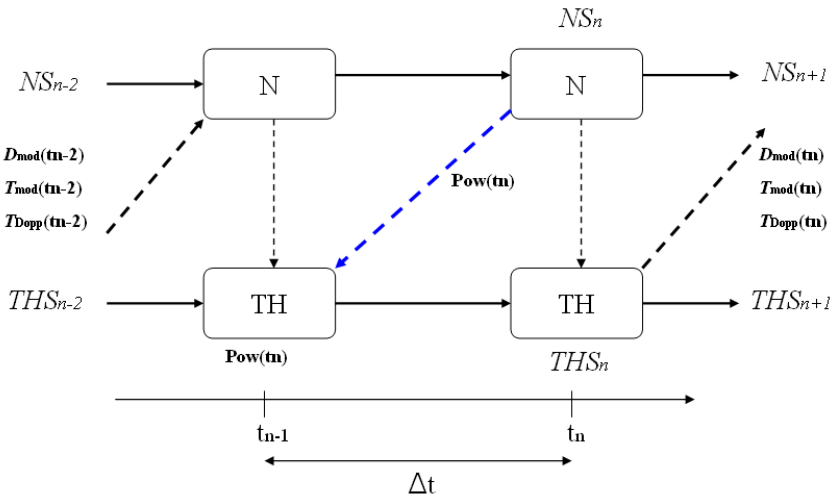
Parallel coupling: The parallel type of coupling refers to a combined system, at which the core N(core model) and TH(core model) are solved with one solver, but the boundary conditions provided by the system code are first used to solve the TH(coolant system) of the plant.

[1] Refer to the report CTH-NT-288, Figure 8, 9 and 10.

3.1.2 Feedback management, interaction between domains and time advancement

Operator Splitting coupling: This type of explicit coupling strategy is based on the coupling of existing mono-disciplinary models. As can be seen in figure 2, the N solution (NS) and the TH solution (THS) between two consecutive time steps are given by equations 7 and 8 respectively;

Figure 2: Operator Splitting coupling approach [4].



$$NS_{t=n} = N(\Delta t, NS_{t=n-1}, THS_{t=n-1}), \tag{7}$$

$$THS_{t=n} = TH(\Delta t, NS_{t=n}, THS_{t=n-1}). \tag{8}$$

In the previous equations, $NS_{t=n}$ is the neutronic solution during a specific time interval, and which is function of the time step, and the NS and THS solutions at the previous time step ($t = n - 1$). The thermal-hydraulic solution during the same time interval, $THS_{t=n}$, is also function of the time step, the NS at the current time step, and the THS at a previous time step. The blue arrow in figure 2 indicates only the code (solver) that is called next. In this case, the N solution ends and the TH solution follows.

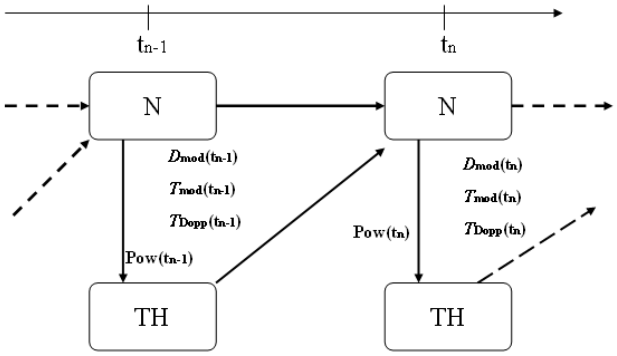
Multiple time steps marching schemes allow the solution of a selected code to proceed with several steps, while the other coupled code marches only with one large step. This temporal adaptive algorithm was developed to perform the synchronization and optimization of the performance of 3D N/TH fuel assembly and sub-channel analysis coupled code systems. However, it was noted [2] that applying automatic time steps selection to some models, did not increase the qualitative precision of the solution compared to fixed time steps implicit schemes, but it reduces significantly the computational cost. Coming back to the staggered OS [5], the convergence of the solution is ruled by the time step where in general a master code controls the time advancement. Small time steps are required specifically for the N (e.g. very small neutron generation time in the order of t = 10.0E-4 s). Examples of this type of coupling which involve OS schemes are TRACE/PARCS, RELAP5/PARCS and CRONOS2/FLICA4. The general idea is that the coupling is based on iterative schemes where one code provides the boundary conditions to the second code and so on until the last code of the simulation system completes one overall temporal step.

Semi-implicit coupling: Semi-implicit methods, highlighted in figure 3, have the advantage of employing feedback parameters from old and new time steps (as done in TRAC-PF1/NEM and SIMTRAN). This might be of specific interest when dealing with power maps as done with SUBCHANFLOW. The main disadvantage of this method is its instability caused by the non-converged feedback parameters similar to the OS approach. Equations 9 and 10 describe this N/TH feedback process.

$$NS_{\Delta t} = N(\Delta t, N_{n-1}, TH_{n-1}), \tag{9}$$

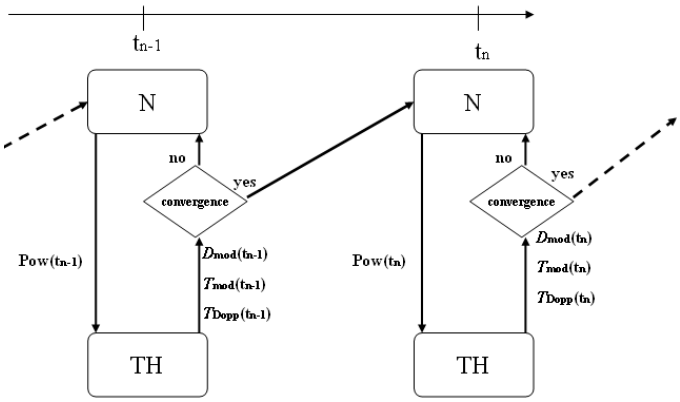
$$THS_{\Delta t} = TH(\Delta t, N_n, TH_n). \tag{10}$$

Figure 3: Semi-implicit coupling approach [4].



Implicit coupling: In the implicit approach (Figure 4), convergence of the individual codes and of the feedback is required. This method has the advantage of being the most accurate and stable out of the others. Several different approaches to solve the implicit coupling involve JFNK methods [6] (which will be described in more detail in the following sections), where improvements in the areas of convergence, dynamic time steps and non-linearities are envisaged. As before, equations 11 and 14 describe this approach.

Figure 4: Implicit coupling approach [4].



$$NS_{\Delta t} = N(\Delta t, N_{n-1,n}, TH_n), \tag{11}$$

$$THS_{\Delta t} = TH(\Delta t, N_{n-1,n}, TH_n). \tag{12}$$

Fixed point iteration (Picard) :

The Fixed Point Iteration (FPI) scheme is not an implicit scheme but approximates it by adding an iteration loop to the current explicit scheme. The advantage of the method is that it allows the use of larger time steps; however the nested loop iteration could take much more time in getting a converged solution and could be less efficient than the explicit scheme with small time steps. This type of coupling is implemented in DYNSUB[2]. Additional modifications could accelerate the rate of convergence of the non-linear Picard iterations in order to make it a viable candidate for reactor analysis. Schemes such as Steffensen and vector Wynn-Epsilon algorithms [5] can be used to accelerate the convergence rate.

$$NS_{\Delta t} = N(\Delta t, N_{n-1,n}, TH_n), \tag{13}$$

$$THS_{\Delta t} = TH(\Delta t, N_n, TH_n). \tag{14}$$

3.2 Jacobian-Free Newton-Krylov

The JFNK method [7] combined with physics-based preconditioning is understood to be a modern multi-physics algorithm. It significantly differs from traditional OS methods. The method has been widely used and its objective lies on obtaining a final solution for non-linear solvers by treating all equations simultaneously. This method presents several advantages and disadvantages, and they are based on a nonlinear iterative method (Newton) applied to a residual equation of all the sets of multi-physics PDEs, in which each Newton step is solved iteratively using a Krylov method.

Advantages:

1. It performs Newton iterations on a complicated multi-scale function, where forming the Jacobian is complicated.
2. No splitting or linearization error occurs.
3. It is a clean way to include a variety of non-linear phenomena.
4. One can implement a variety of higher order time discretization.
5. It opens the door for sensitivity analysis, compared to other numerical techniques.
6. The overall non-linear convergence of the method is not directly affected by the approximations made in the preconditioning.
7. It allows Solving the full dimensional system implicitly.

Disadvantages:

1. One needs to solve non-linear problems with iterations (Newton method) and a good initial guess is needed.
2. One must produce effective pre-conditioners.
3. Convergence issues near local extrema exists.

Solving nonlinear PDE's in a tightly coupled manner has its own set of challenges. One is the need to form a large linear system for use during Newton iterations. The JFNK is based on a nonlinear iterative method applied to a residual equation of the set of multi-physics PDEs, in which each Newton step is solved iteratively using a linear decomposition on a Krylov vector basis. As the number of solution variables grows, the matrix holding the Jacobian entries also grows. The increasing size of the matrix

[2]Refer to report no. CTH-NT-288, section 2.2.1, where details about DYNSUB are given in addition to the representative figure 4.

means a greater memory consumption. Further, it takes time to fill in the full Jacobian and for problems with highly nonlinear material properties the true Jacobian can often be difficult to obtain. For these reasons, Krylov methods for solving the resultant linear systems obtained by the Newton linearization are used. The Krylov method, requires robust techniques to approach the solution of some coupled multi-physics problems which produce linear systems that are block un-symmetric. Some of these methods involve the GMRES (Generalized Minimum Residual), BiCGSTAB (Bi-Conjugated Gradient Stabilized) and TFQMR (Transpose-free Quasi Minimal Residual). The first one mentioned is the most popular due to its efficiency in solving non-symmetric system of equations, which makes it attractive for the usage in tightly coupled multi-physics systems.

This iterative methods for solving the systems require only a matrix-vector product and not the full matrix (do not require the Jacobian matrix itself but simply the action of the Jacobian matrix on a vector). Therefore, the full Jacobian is not necessary and the Jacobian-free scheme is utilized. Approximating this matrix-vector product by differencing, which requires two nonlinear function evaluations, is the basis of the JFNK method (only requires residual evaluations). This allows for a modular, pluggable architecture that greatly simplifies the addition of new physics and coupling them together. By separating the physics processes into kernels (i.e. N and TH), each kernel is responsible for evaluating its portion of non-linear residual. Hereafter more details are given about Newton's method, specially the steps needed for applying the method which involve:

1. Form the Jacobian matrix.
2. Solve the sparse linear system.
3. Apply this update to obtain the next iteration of the solution state vector.

The Newton iteration starts with the residual equation 15;

$$\frac{\partial u}{\partial t} - f(u,t) = res(u) = 0, \quad u = [u_1, ..., u_m]^T, \tag{15}$$

where "res" is the function obtained by means of the governing equations of the system and "u" is the state vector. Taking the Taylor series expansion of the function, neglecting higher-order terms and setting $res(u^{k+1})$ equal to the zero vector, a set of equations of the form $A\hat{x} = b$ is obtained, such as in equations 16 and 17;

$$J(u^k)\delta u^k = -res(u^k), \approx A\hat{x} = b, \tag{16}$$

The Newton step is updated as:

$$u^{k+1} = u^k + \delta u^k, \tag{17}$$

where δu^k is the updated vector and "k" is the iteration index. The residual vector of each system may be computed for an approximated solution $\hat{x}$ as in equations 18 and 19;

$$res = b - A\hat{x}, \tag{18}$$

$$res_i \equiv \sum_j [y_i - f(x_j, ..., x_N)]^2. \tag{19}$$

Moreover, in vector notation the $(i,j)^{th}$ element of the Jacobian matrix is given by $J_{i,j} = \frac{\partial res_i(u)}{\partial u_j}$. Then, the global function res can be made by concatenating the evaluations of the different systems. If one does not know a solution "x" but an approximation "$\hat{x}$", the residual (res) can be computed but not the error. In many cases, the smallest the residual is, the closer the approximation is to the solution.

This Newton iterative scheme is formulated to reduce the residual. But the matrix-vector products $J(u^k)\delta u^k$ (δu^k is the perturbation between Newton iterations, as in equation 17) are approximated by the finite-difference form as the directional Gateaux derivative. One can see the advantage of the JFNK, which lies on the determination of the approximated update vector and which does not require computing or inverting the Jacobian. Only the action of the Jacobian on the Krylov vector is needed. In order to

accelerate the convergence of the method, a preconditioning "M"is applied (noting that the nonlinear iteration index "k" can be dropped out since the Krylov iteration is performed at a fixed "k"):

$$J(u)M^{-1}\delta u = \frac{res(u + hM^{-1}\delta u) - res(u)}{h}. \tag{20}$$

Where in equation 20, *res(u)* is the non-linear residual function, h is the perturbation parameter and M symbolically represents the preconditioning matrix. In fact, all Kernels are required to supply a residual. Alternatively, the residual could be calculated by calling a third party application or doing some calculations on the meso-scale and extrapolating to form the engineering scale residual. Kernels can also optionally provide a Jacobian or preconditioning matrix, which can then be used for physics based preconditioning of the matrix free calculation. Often, the diagonal block is provided by a Kernel which is usually sufficient for effective preconditioning (necessary for optimal convergence of the Krylov solver). In PARCS for instance, the Krylov subspace method BICGSTAB preconditioned with BILU3D is employed to solve the CMFD problem.

Once again, in the JFNK method, the nonlinear iteration is not solved directly and the Jacobian is neither formed nor inverted at each iteration step. Instead, the update δu^k vector is projected on a Krylov subspace where the projected vector is iteratively solved. The approximated solution at the l- th Krylov iteration, δu_l, is constructed through a linear combination of the Krylov vectors (the k-eigenvalue problem could become part of the Krylov solution vector). The solver approximates the update vector on a Krylov subspace of dimension l as shown in equation 21.

$$\delta u_l^k = \delta u_0^k + \sum_{j=0}^{l-1} \alpha_j (J^k)^j (J^k \delta u_0^k + res(u^k))), \tag{21}$$

with δu_0^k as an initial guess and α_j is a scalar part of the Krylov iteration. In fact, this initial guess is evaluated as: $u_0 = -res_0(u) - J\delta u_0$.

Then, using the selected iterative solver (GMRES or BiCGSTAB for instance) the dimension of the subspace is increased until a good convergence effect on the Newton iteration is predicted, using the inexact Newton method stopping criterion:

$$\parallel J^k \delta u_l^k + res(u^k) \parallel^2 < \varepsilon_L \parallel res(u^k) \parallel^2, \tag{22}$$

In the previous equation, $\varepsilon_L << 1$. The term inexact refers only to the stopping criteria of the inner iterative solver, since this loop is to be re-executed until the outer Newton iteration loop has converged. When using the GMRES as the inner iterative solver, the linearized update vector is constructed from an orthonormal basis of the Krylov subspace generated by, $\delta u_{l+1}^k = J^k \delta u_l^k + res(u^k)$. This is obtained from an Arnoldi procedure which consists of an iterative orthogonal decomposition of $\delta u_{l+1}^k / \parallel \delta u_{l+1}^k \parallel$. This results in an orthogonal basis on which an update vector can be defined, and whose coefficients are then computed to minimize (in the least-square sense) the residual $\parallel J^k \delta u_l^k + res(u^k) \parallel^2$.

The determination of an efficient set of convergence parameters is essential to the success of the JFNK method. Too large values would result in more non-linear Newton iterations and thus larger CPU consumption, whereas a too small value would increase the number of linear iterations, hence enlarging the size or the number of Krylov vectors of the problem. One should also consider an additional parameter λ, which is sometimes used to damp the update predicted by the Newton, i.e. $u^{k+1} = u^k + \lambda \delta u^k$, with $0 \leq \lambda \leq 1$. One of the reasons for this is that Newton methods may not converge or converge very slowly if the initial guess is too far from the solution.

3.2.1 Preconditioning the JFNK:

The objective of preconditioning the JFNK is to reduce the number or Krylov iterations (related) to convergence, by efficiently clustering eigenvalues of the iteration matrix. This iteration matrix should be a good approximation of the Jacobian and should be easier to form and solve as compared to the Jacobian matrix itself. To be applicable to large problems, JFNK methods require a proper preconditioner to reduce the number of the solver iterations. A preconditioning consists of formally applying an operator (M in equation 20) in order to reduce the condition number (i.e. the ratio of the largest eigenvalue to the lowest eigenvalue) for the system, which helps improving the convergence of the iterative method. Since applying left preconditioners changes the norm of the residual by which convergence to a linear iterative method is measured, right preconditioning is preferred. From equation 20, it is seen that the best preconditioner operator would be Jacobian itself, since the operator to be inverted would become the identity matrix. This is the reason why some Newton-Krylov methods do form and store full or partial explicit Jacobians during the procedure, which increase the size of the computation. The quality of the preconditioner will be ultimately measured in the reduction of the Krylov iterations compared to the additional computation required to calculate the action of the preconditioner, M^{-1}. Some preconditioning approaches are standard, which involve Incomplete Cholesky factorization, Incomplete-LU factorization, Sparse Approximate Inverses, Block-Jacobi splitting, additive Schwarz method, Newton-Krylov-Schwarz, and physics-based preconditioning. The basic principles of behind preconditioning are mentioned hereafter. The fundamental idea is to use a Krylov subspace method on a modified system that satisfies:

$$M^{-1}Ax = M^{-1}b, \tag{23}$$

where the matrix product "$M^{-1}A$" does not need to be formulated. The equation 23 represents in fact the left preconditioning procedure. Right preconditioning involves:

$AM^{-1}u = b$, with $x = M^{-1}u$,

and the split preconditioning (central) assumes "$M = M_{left}M_{right}$" satisfying:

$M_{left}^{-1}AM_{right}^{-1}u = M_{left}^{-1}b$, with $x = M_{right}^{-1}u$,

and the general Krylov algorithm for this preconditioner type is:

Algorithm 1 Illustration of the Krylov algorithm applied to the split type of preconditioning.

. x_0=initial guess
. $r_0 = b - Ax_0$
. Perform k iteration over the Krylov space method for:
. $M_{left}^{-1}AM_{right}^{-1}u = M_{left}^{-1}r_0$
. with $u_0 = 0$.
. $x_k = x_0 + M_{right}^{-1}u_k$

In fact, iterative techniques such as Block-SOR or Multi-grid can be used as preconditioners. Based on equations 16 to 23, specific algorithms for preconditioning can be formulated [3]. The preconditioner should satisfy some requirements and characteristics:

1. Convergence should be faster for preconditioned systems than for the original system.
2. Operation with the preconditioner should be easy to perform.
3. Left preconditioning does not require extra steps to compute u.

[3]The objective of this report is not to give a detailed description of these type of algorithms (detailed formulation of these methods is given in [8] and [9] among many other references).

4. Right preconditioning does not affect the residual norm.
5. Central preconditioning preserves symmetry.

Preconditioned Conjugate Gradient (PCG).

Algorithm 2 Illustration of the algorithm applied to PCG.

Compute $r_0 = b - Ax_0, z_0 = M^{-1}r_0$ and $p_0 = z_0$
. while convergence
. For j = 0,1...
. $a_j = (r_j, z_j)/(Ap_j, p_j)$
. $x_{j+1} = x_j + a_j p_j$
. $r_{j+1} = r_j - a_j Ap_j$
. $z_{j+1} = M^{-1}r_{j+1}$
. $\beta_j = (r_{j+1}, z_{j+1})/(r_j, z_j)$
. $p_{j+1} = z_{j+1} + \beta_j p_j$
. end
. end

Conjugate Gradient with Split.

Algorithm 3 Illustration of the algorithm applied to conjugate gradient with split preconditioning.

Compute $r_0 = b - Ax_0, z_0 = M^{-1}r_0$ and $p_0 = M^{-T}z_0$
. while convergence
. For j = 0,1...
. $a_j = (z_j, z_j)/(Ap_j, p_j)$
. $x_{j+1} = x_j + a_j p_j$
. $z_{j+1} = z_j - a_j M^{-1}Ap_j$
. $\beta_j = (z_{j+1}, z_{j+1})/(z_j, z_j)$
. $p_{j+1} = M^{-T}z_{j+1} + \beta_j p_j$
. end
. end

The GMRES with no preconditioning.

Algorithm 4 Illustration of the algorithm applied to GMRES with no preconditioning.

Select x_0 and a Krylov space size m
Compute $r_0 = b - Ax_0, \beta = \| r_0 \|_2$ and $v_1 = \frac{r_0}{\beta}$
. For j = 0,1...m
. Compute $w = Av_j$
. for i = 1...j, do: $h_{i,j} = (w, v_i)$ and $w = w - h_{i,j}v_i$
. $h_{j+1}, 1 = \| w \|_2, v_{j+1} = w/(h_{j+1,j})$
. Define: $V_m = [v_1, ..., v_m]$ and $H_m = h_{i,j}$
. end
. Compute the update solution: $x_m = x_0 + V_m y_m$
. where $y_m = min \| \beta e_1 - H_m y \|_2$ and $e = [1, 0, .., 0]^T$
. If satisfied stop, else restart at the Arnoldi step and set $x_0 < -x_m$

Which differs from the GMRES with right preconditioning.

Algorithm 5 Illustration of the algorithm applied to GMRES with right preconditioning.

Select x_0 and a Krylov space size m
Compute $r_0 = b - Ax_0, \beta = \| r_0 \|_2$ and $v_1 = \frac{r_0}{\beta}$
. For j = 0,1...m
. Compute $z = M^{-1}v_j$)
. Compute $w = Az_j$)
. for i = 1...j, do: $h_{i,j} = (w, v_i)$ and $w = w - h_{i,j}v_i$
. $h_{j+1}, 1 = \| w \|_2, v_{j+1} = w/(h_{j+1,j})$
. Define: $V_m = [v_1, ..., v_m]$ and $H_m = h_{i,j}$
. end
. Compute the update solution: $x_m = x_0 + M^{-1}V_m y_m$
. where $y_m = min \| \beta e_1 - H_m y \|_2$ and $e = [1, 0, .., 0]^T$
. If satisfied stop, else restart at the Arnoldi step and set $x_0 < -x_m$

In addition, the preconditioned BiCGSTAB algorithm is shown in the following algorithm.

Algorithm 6 Preconditioned BiCGSTAB algorithm [8].

. $r_0 = b - Ax_0$ for an initial guess x_o
. $\rho_0 = \alpha = \omega_0 = 1$
. $v_0 = p_0 = 0$
. For i = 1,2,3...
. $\rho_i = (r_0, r_{i-1})$
. $\beta = (\rho_i/\rho_{i-1}) * (\alpha/\omega_{i-1})$
. $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1}v_{i-1})$
. Solve $Py = p_i$ for y
. $v_i = Ay$
. $\alpha = \rho_i/(r_0, vi)$
. $s = r_{i-1} - \alpha V_i$
. Solve $Pz = s$ for z
. $t = Az$. $\omega_i = (t, s)/(t, t)$. $x_i = x_{i-1} + \alpha y + \omega_i z$. if x_i is converged, then quit. . $r_i = s - \omega_i t$. end

Physics-based preconditioning: The physics based preconditioning will be addressed now and in fact (for the sake of exemplification), if one considers a linear thermo-mechanical system depending on temperature (T) and x,y,z displacements (coupled to each other as well as being coupled to the temperature), an ideal preconditioning matrix could then look like 24 [10];

$$M = \begin{bmatrix} (F_T)_t & 0 & 0 & 0 \\ (F_x)_t & (F_x)_x & (F_x)_y & (F_x)_z \\ (F_y)_t & (F_y)_x & (F_y)_y & (F_y)_z \\ (F_z)_t & (F_z)_x & (F_z)_y & (F_z)_z \end{bmatrix}. \tag{24}$$

In this case "F" is the function relating the temperature and space variables. The solution of the system will proceed in rows: the result of the first row will be used to solve for the next row and so on.

This class of preconditioners of the JFNK method tend to reduce significantly the implicit system, or a sequence of explicit or implicit systems may be solved in place of the fully coupled system. The idea is to implement multiple iterations of the "delta form" algorithm as preconditioner. By "delta form", it is implied that the equations follow a trend similar to the one discussed previously, where the linear system $Pu = b$ is solved for δu as $P\delta u = b - P\delta u_0$, which is actually referred to as the residual form. To construct the preconditioners, the following steps should be applied.

1. Construct the root finding form of the system of PDEs.
2. A discretized method is constructed by implementing the correct linearized and discretization techniques.
3. Solve the system for the desired solution and apply the residuals.

It is to note that since JFNK does not need the creation of the Jacobian, time-splitting approaches, such as semi-implicit methods, can be used as preconditioners. The inverse of the Jacobian can be viewed as a linearized time-step advancement scheme. Thus, a less computationally expensive time-step advancement scheme could be used as the preconditioner, with the difference that this scheme would be applied to iteratively update vectors rather than the residual. For instance, the so called implicit balance solution, consists of nesting a fast but inaccurate solution method (Operator Split Semi-Implicit, OSSI) as a preconditioner of a more accurate JFNK method in order to provide a solution method which is both fast and accurate. The resulting preconditioned JFNK method can be written as:

for each time step
—-for each Newton iteration
———for each Krylov iteration
—————COMPUTE an OSSI and calculate the residual differences
———end Krylov iteration
—-end Newton iteration
end time step

In the situation where the coupling is to be based on an existing set of codes, the implementation of such a method could require significant modifications to the structure of the code, unless their solvers were already based on JFNK methods. A solution is to use an Approximated Block Newton (ABN) method [11], which is based on a different approach for seeking modularity and simpler implementation, where the different solvers of the coupling can be preserved as black-box solvers. The original idea is to derive a Newton method based not on the original equation set, but on the solvers or each sub-problem.

To understand the ABN method, one can consider two solvers, F and G, possibly implicit in time with variables x and y. The coupled system can be formulated as in equations 25 and 26:

$$x_{k+1}^{n+1} = F(x_k^{n+1}, y), \tag{25}$$

$$y_{k+1}^{n+1} = G(x, y_k^{n+1}). \tag{26}$$

Here k is counting the inner iterations of the two solvers, which can be only one if the codes solve their temporal step directly. The problem can be written in a residual form as in equations 27 and 28:

$$f(x, y) = x - F(x, y) = 0, \tag{27}$$

$$g(x, y) \equiv y - G(x, y) = 0, \tag{28}$$

and a block Newton method is constructed as in 29;

$$\begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} f \\ g \end{pmatrix}. \tag{29}$$

The idea is to left-precondition the system using the Jacobian inverses of the sub-problem, by proving

that there is no need to compute the Jacobian blocks, as in equation 30.

$$\begin{pmatrix} \frac{\partial f}{\partial x}^{-1} & 0 \\ 0 & \frac{\partial g}{\partial y}^{-1} \end{pmatrix} \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} \frac{\partial f}{\partial x}^{-1} & 0 \\ 0 & \frac{\partial g}{\partial y}^{-1} \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix}. \tag{30}$$

Using Gauss elimination the previous system can be formulated as in 31:

$$\begin{pmatrix} I & C \\ 0 & S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} q \\ r \end{pmatrix}. \tag{31}$$

with $C = (\frac{\partial f}{\partial x})^{-1} \frac{\partial f}{\partial y}; S = I - (\frac{\partial g}{\partial y})^{-1} \frac{\partial g}{\partial x} C; q = (\frac{\partial f}{\partial x})^{-1} f$ and $r = (\frac{\partial g}{\partial y})^{-1} g - (\frac{\partial g}{\partial y})^{-1} \frac{\partial g}{\partial x} q$.

The update for Δy can be solved separately and can be then included in the equation for Δx to solve the entire system at each Newton iteration. The expressions for C, S and r include partial derivatives (coupling derivatives) which cannot be obtained easily with black box solvers. However, we can do:

1. Taylor expansion that include the "coupling derivatives".
2. So $S\Delta y = -r$ is not solved for Δy directly but using iterative methods, i.e. Newton-Krylov method.

The Newton-Krylov method requires only the action of S and C on the Krylov vectors. Second, using a Taylor expansion of the 1st order to express the "coupling derivative" terms of S and C, the action of them on a Krylov vector v can be approximated by equations 32, 33 and 34 as;

$$Cv = (\frac{\partial f}{\partial x})^{-1} \frac{\partial f}{\partial y} v \approx \frac{1}{\varepsilon} (\frac{\partial f}{\partial x})^{-1} [f(x, y + \varepsilon v) - f(x, y)], \tag{32}$$

$$Sv = v - (\frac{\partial g}{\partial y})^{-1} \frac{\partial g}{\partial x} Cv \approx v + \frac{1}{\varepsilon} (\frac{\partial g}{\partial y})^{-1} [g(x, y - \varepsilon Cv) - g(x, y)], \tag{33}$$

$$r = (\frac{\partial g}{\partial y})^{-1} (g - \frac{\partial g}{\partial x} q) \approx (\frac{\partial g}{\partial y})^{-1} g(x + q, y). \tag{34}$$

These new expressions include only terms in the form $(\frac{\partial f}{\partial x})^{-1} f$ or $(\frac{\partial g}{\partial y})^{-1} g$, which can be approximated using F and G by considering a block-wise Newton iteration for the system mentioned in equations 27 and 28, but where the variables x and y at iteration k+1 are approximated using equations 25 and 26, by one fixed point iteration of F and G as highlighted below (equations 35 and 36).

$$x_{k+1} = x_k - (\frac{\partial f}{\partial x}(x_k, y))^{-1} f(x_k, y) = F(x_k, y), \tag{35}$$

$$y_{k+1} = y_k - (\frac{\partial g}{\partial x}(x, y_k))^{-1} g(x, y_k) = G(x, y_k). \tag{36}$$

Therefore, the implementation of the Approximated Block Newton algorithm for a transient problem can be decomposed as:

for each time step
—-for each Newton iteration
———-1) Compute $q = x_i - F(x_i, y_i)$.
———-2) Compute $G = G(x_i, y_i)$.
———-3) Compute $r = y_i - G(x_i + q, y_i)$.
———-4) Solve for $S\Delta y = -r$ for Δy for a given tolerance using GMRES method with:
————————- $Cv = -\frac{1}{\varepsilon}[F(x_i, y_i + \varepsilon v) - F(x_i, y_i)]$
————————- $Sv = v - \frac{1}{\varepsilon}[G(x_i - \varepsilon Cv, y_i) - G]$
———-5) Compute $y_{i+1} = y_i + \Delta y$ and $x_{i+1} = F(x_i, y_{i+1})$.
———-6) Convergence check.
—-end Newton iteration
end time step

The advantage of ABN methods is that the large system is broken down into smaller blocks, which results in inverting a matrix that is smaller than in regular JFNK methods. Like for any Newton method, globalization and optimization techniques could help to enlarge the radius of convergence and the robustness of the system.

3.3 Other techniques for solving non-linear coupled systems

Three specific solvers will be discussed, the PETSc, UMFPACK and MAREID (the last two being actually incorporated as optional solvers in the POLCA-T code).

PETSc [12]:

The Portable Extensible Toolkit for Scientific Computation (PETSc) is suitable for solving large-scale scientific application codes in FORTRAN, C, C++, and Python. The software comprises a powerful set of tools for numerical solutions of partial differential equations and related problems on high-performance computers. PETSc is also callable from MATLAB. This software consists of a variety of libraries, each which manipulates a particular family of objects (i.e. vectors) and the operations one would like to perform on the objects. Some modules of PETSc deal with:

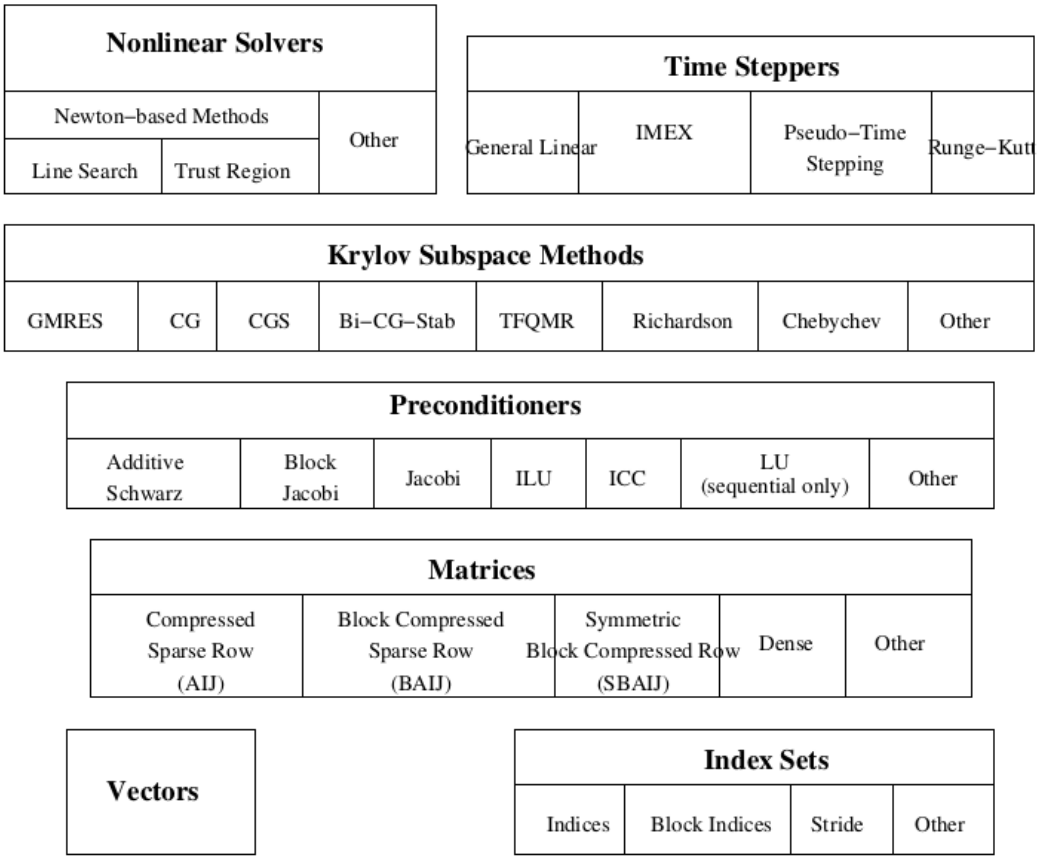
- Index sets, including permutations, renumbering, etc...
- Vectors, matrices.
- Managing interactions between mesh data structures and vectors and matrices.
- Over fifteen Krylov subspace methods.
- Many preconditioners, including multi-grid, block solvers, and sparse direct solvers.
- Non-linear solvers.
- Time steppers for solving time-dependent non-linear PDEs.

The PETSc enables easy comparison and use of different algorithms, i.e. it is straightforward to experiment with different Krylov subspace methods, preconditioners, or truncated Newton methods. Figure 5 presents several of the individual parts in more detail.

UMFPACK [13]:

UMFPACK is a set of routines for solving systems of linear equations, $Ax = b$, when $\mathbf{A}$ is sparse and unsymmetrical. The sparse matrix $\mathbf{A}$ can be square or rectangular, singular or non-singular, and real or complex, or any combination. UMFPACK is a built-in routine in MATLAB used by the forward and backslash operator, and the *lu* routine. Five primary UMFPACK routines are required to factorize $\mathbf{A}$ or solve for $Ax = b$.

Figure 5: Numerical libraries of PETSc [12].



- umfpack_di_symbolic: Returns an opaque Symbolic object as a void pointer.
- umfpack_di_numeric: Returns an opaque Numeric object as a void pointer. The object contains the numerical factorization and is used by the following routine.
- umfpack_di_solve: Solves a sparse linear system $Ax = b, A^T x = b$ or systems involving just **L** or **U**, using the numeric factorization computed by the previous routine.
- umfpack_di_free_symbolic: Frees the Symbolic object created before.
- umfpack_di_free_numeric: Frees the Numeric object created before.

UMFPACK first finds a column pre-ordering that reduces fill-in, without regard to numerical values. It scales and analyzes the matrix and then automatically selects one of the three strategies for pre-ordering the rows and columns: *unsymmetrical 2-by-2* and *symmetric*. Several rules apply, and the first one that matches defines the strategy. Once the strategy is selected, the factorization of the matrix **A** is broken down into the factorization of a sequence of dense rectangular frontal matrices. The frontal matrices are related to each other by a super-nodal column elimination tree, in which each node in the tree represents one frontal matrix. This analysis phase also determines upper bounds on the memory usage, the floating-point operation count, and the number of non-zeros in the LU factors.

MAREID [14]:

MAREID is based on MA28 (ABC) package (direct solver for sparse systems of linear equations). The package solves the system $Ax = b$ or optionally $A^T x = b$. It will solve another system having the same sparsity pattern taking much less processing time. The method is a variant of Gaussian elimination for sparse systems. In addition, there are four entries as argument lists and calling sequences:

- MA28A: Decomposes **A** into factors using a pivotal strategy designed to compromise between maintaining sparsity and controlling loss of accuracy through roundoff.
- MA28B: Factorizes a new matrix **A** of the same pattern, using the pivotal sequence determined by an earlier entry to MA28A. It also permits non-zeros in positions that filled in during the previous factorization.
- MA28C: Uses the factors produced by MA28A or MA28B to solve $Ax = b$ or optionally $A^T x = b$.

Finally, the reader should know that there are different types of techniques to solve systems of coupled non-linear PDEs. In addition, acceleration techniques for the Newton type of solution algorithms and which involve Krylov subspaces concern Anderson and Broyden mixing techniques [15]. Some of these Newton types of techniques are:

- Trust region methods.
- Broyden method.
- Secant method.
- Halley method.
- Bächlund transformation method.
- Inverse scattering transformation method.
- Homotopy perturbation method.
- Lindstedt-Pointcare methods.
- Adomian decomposition method.
- F-expansion method.
- Jacobi elliptic function expansion method.
- Complex hyperbolic function method.
- Exp-function method.
- Multi-objective optimization problem method.
- Galerkin method.
- Muller's method (polynomial approximation).

4 Conclusions

Due to the strong coupling between the neutronic and thermal-hydraulics system of equations (and physics), a need to better understand this coupling is an unavoidable task. The type of interaction between codes (feedback exchange and time advancement) is an critical issue to be considered while doing coupled calculations in order to obtain accurate results within a reasonable computational time. A very consistent and broad literature review is presented in this report, which involves different types of coupling schemes and the Jacobian-Free-Newton Krylov method.

Generally two types of schemes are widely used, the explicit and implicit type of schemes. Both present several advantages and disadvantages.

Advantages:

- Explicit
  Easy to implement and parallelize, low cost per time step.
  Good starting point for the development of CFD softwares (generally complex softwares).
- Implicit
  Stable over a wide range of time steps, sometimes unconditionally stable.
  Constitute excellent iterative solvers for steady-state problems.

Disadvantages:

- Explicit

  Small time steps are required for stability reasons, especially if the velocity and/or mesh size are varying strongly

  Extremely inefficient for solution of stationary problems unless local time-stepping is employed.

- Implicit

  Difficult to implement and parallelize, high cost per time step, insufficiently accurate for truly transient problems at large Δt.

  Convergence of linear solvers deteriorates/fails as Δt increases.

Further work involves the development or use of a simple tool to apply the JFNK method to simple core geometries and simplifications. This could be a good starting point to understand its behavior and optimize its convergence by implementing the right preconditioners. In addition, issues on Anderson Mixing and Broyden Mixing for accelerating the JFNK convergence are envisaged. Finally, the reader should be aware that a companion report "Survey of the different numerical algorithms used in present computer codes" exists and more details about coupling techniques are presented there.

5 Acknowledgements

Spacial thanks are extended to U. Bredolt and E. Muller from Westinghouse Sweden AB, for providing important scientific information concerning the project. The author is thankful to the division of Nuclear Engineering, part of the department of Applied Physics at the Chalmers University of Technology, for providing a good environment and facilitating the completion of this task. This work has been financed and supported by the Nordic Thermal-Hydraulic Network (NORTHNET) with research contracts: 4500297145 (Forsmark Kraftgrupp AB), 634282-052, (Ringhals AB) 4500056499/Q19/0203, (Oskarsharm Kraftgrupp AB), SSM2013-2195, (Stålsäkerhetsnuydigheter), and 4500609943 (Westinghouse Electric Sweden AB).

References

[1] D. Gaston et al. "Tightly Coupled multi-physics simulations for pebble bed reactors". In: *Mathematics and Computation, New York*. 2009.

[2] M. Grandi. "Effects of the discretization and neutronic thermal-hydraulic coupling on LWR transients". In: *NURETH-13 Japan*. 2009.

[3] J. Jeong et al. "A Multi-Dimensional Theraml-Hydraulic System Analysis Code: MARS 1.3.1". In: *Journal of Korean Nuclear Society* na (1999), na.

[4] M. Calleja. "Improvements in Multi-physics and Multi-scale Methodologies for Safety-related investigatinos of Pressurized Water Reactors within the NURESIM simulation platform". PhD thesis. Karlsruhe Institute of Technology, 2013.

[5] O. Zerkak et al. "Revising temporal accuracy in neutronics/thermal-hydraulics code coupling using the NURESIM LWR simulation platform". In: *NURETH-14, Toronto*. 2011.

[6] V. Mousseau and M. Pope. "Accuracy and efficiency of a coupled neutronic and thermalhydraulic model". In: *Nuclear Engineering and Design* 41 (2009), na.

[7] W.J. Garland and B.J. Hand. "Simple Functions for the Fast Approximation of Light Warter Thermodynamic Properties". In: *Nuclear Engineering and Design* 113 (1998), pp. 21–34.

[8] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Tech. rep. PWS Publishing Company, 1996.

[9] Y. Saad. *Numerical Methods for Large Eigenvalue problems*. Ed. by SIAM. Manchester University Press, 2011.

[10] D. Gaston, G. Hasen, and C. Newman. "MOOSE: A parallel computational framework for coupled systems of nonlinear equations." In: *International Conference on Mathematics, Computational Methods and Reactor Physics*. 2009.

[11] A. Yeckel, L. Lun, and J. Derby. "An approximate block Newton method for coupled iterations of nunlinear solvers: Theory and conjugate heat transfer applications". In: *Journal of Computational Physics* 228 (2009), pp. 8566–8588.

[12] S. Balay, J. Brown, and K. Cuschelman. *PETSc Users Manual. Revision 3.4*. Mathematics and Computer Science Division, Argonne National Laboratory. 2013.

[13] T. Davis. *UMFPACK Version 5.2.0 User Guide & UMFPACK Version 5.2 Quick Start Guide*. Department of Computer, Information Science, and Engineering. University of Florida. 2007.

[14] I. Duff. *MA28: a set of Fortran subroutines for sparse unsymmetric linear equations*. Tech. rep. Mathematical Physics and Mathematics - Report no. AERE-R-8730-REV, 1980.

[15] H. Walker and P. Ni. "ANDERSON ACCELERATION FOR FIXED POINT ITERATIONS". In: *Society for Industrial and Applied Mathematics* 49 (2011), pp. 1712–1735.