Krist V. Gernaey, Jakob K. Huusom and Rafiqul Gani (Eds.), 12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering.
31 May - 4 June 2015, Copenhagen, Denmark. © 2015 Elsevier B.V. All rights reserved.

Degeneracy Hunter: An Algorithm for Determining Irreducible Sets of Degenerate Constraints in Mathematical Programs

Alexander W. Dowling[a] and Lorenz T. Biegler[a]

[a]*Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, United States*
biegler@cmu.edu

Abstract

Degenerate constraints, i.e. constraints that violate the Linearly Independent Constraint Qualification (LICQ), are prevalent in many process optimization problems. These result from poor model formulations (typically human error) and overspecifications, zero flowrates and disappearing units, and recycle loops. Because degenerate constraints lead to singular Karush-Kuhn-Tucker (KKT) systems and significant challenges for numeric solvers, these constraints complicate the solution procedure for nonlinear programs (NLPs, i.e. nonlinear optimization problems). Although most modern NLP solvers implement counter-measures to detect and eliminate degeneracies, increased computational effort and convergence failures may still result. Instead, the best approach is to reformulate the original NLP model. Unfortunately, this is difficult for complex models with thousands of equations. This work describes the *Degeneracy Hunter*, an algorithm that systematically analyzes any iteration from a continuous mathematical program solver and determines irreducible sets of degenerate constraints. This tool allows the expert modeler to focus on only a handful of equations, instead of the thousands that make up a typical process optimization problem. The algorithm has been prototyped in MATLAB and analyzes derivative information exported from GAMS. Calculation of irreducible sets of degenerate equations is formulated as a mixed integer linear program. The algorithm has been applied to a non-convex, highly nonlinear Air Separation Unit (ASU) design problem with 15,000+ variables and constraints, which identified three sources of degenerate equations. Straightforward revisions of the model to remove these degenerate constraints resulted in a 16% decrease in average CPU time and less frequent termination at infeasible points. Identification of these degeneracies would have been virtually impossible without a systematic approach.

Keywords: degenerate constraints, linearly dependent constraints, nonlinear programming

1. Introduction and Motivating Examples

We consider two classes of degenerate constraints: local and global. Local (or point) degeneracies occur only at specific values for variables in the mathematical program. For example, mass balance equations, shown in (1), become degenerate when flowrates go to zero ($F = L = V = 0$). The Jacobian of the equations is shown in (2), with F and z fixed. At the zero flowrate point, pivoting on the first two columns in (2b) shows that the Jacobian matrix is rank deficient.

In contrast, global degeneracies always lead to rank deficient Jacobians, regardless of the value of the variables. These most commonly are a consequence of overspecifications. For example, if $\sum_i y_i = 1$ and $y_i = K_i x_i \; \forall i$ are added to (1), the systems is always overspecified ($2+2n$ variables and $3+2n$ equations when F, z and K are specified) and the Jacobian is rank deficient.

$$F = V + L \tag{1a}$$

$$F z_i = x_i L + y_i V, \quad \forall i \in \{Comps\} \tag{1b}$$

$$\sum_{i \in \{Comps\}} (x_i - y_i) = 0 \tag{1c}$$

$$A = \begin{array}{l} \\ \text{Overall MB} \\ \text{MB } \; i=1 \\ \vdots \\ \text{MB } \; i=n \\ \Sigma \end{array} \begin{array}{c} \begin{array}{cccccccc} L & V & x_1 & \cdots & x_n & y_1 & \cdots & y_n \end{array} \\ \begin{pmatrix} -1 & -1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ -x_1 & -y_1 & -L & \cdots & 0 & -V & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -x_n & -y_n & 0 & \cdots & -L & 0 & \cdots & -V \\ 0 & 0 & 1 & \cdots & 1 & -1 & \cdots & -1 \end{pmatrix} \end{array} \tag{2a}$$

$$A(F = L = V = 0) = \begin{pmatrix} -1 & -1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ -x_1 & -y_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -x_n & -y_n & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 1 & -1 & \cdots & -1 \end{pmatrix} \tag{2b}$$

As another example, consider the system shown in Figure 1, comprising two vessels, a splitter and seven streams. Stream 1 is fed into the first vessel, which has two outlets: streams 2 and 3. Without loss of generality, let streams 2 and 3 represent two different phases in equilibrium. These streams are fed into the second vessel, producing effluent streams 4 and 5. Stream 5 is split into streams 6 and 7, and the latter is recycled into the first vessel. Pressure relationships for these streams are shown in (3). The equations are derived from a two simple rules: streams leaving a vessel are in pressure equilibrium, and pressure cannot increase across vessels and is constant across a splitter. This constraint system is overspecified (and degenerate), as there are seven constraints and seven variables, but there should be one degree of freedom (e.g. P_1 should be free).

$$P_2 = P_3 \tag{3a}$$

$$P_4 = P_5 \tag{3b}$$

$$P_6 = P_7 \tag{3c}$$

$$P_5 = P_7 \tag{3d}$$

$$P_1 \geq P_3 \tag{3e}$$

$$P_2 \geq P_5 \tag{3f}$$

$$P_7 \geq P_3 \tag{3g}$$

Many nonlinear programming (NLP) solvers implement safeguards to mitigate degenerate constraints. For example, CONOPT (Drud, 1994), a large-scale active-set generalized reduced gradient (GRG) optimization solver, removes degenerate constraints from the active set, effectively ignoring these constraints. However, determining the active set for inequality (and degenerate constraints) is an NP-hard task. On the other hand, interior point methods avoid the computational complexity of active set determination by penalizing inequality constraints with a barrier term. As a consequence, degenerate constraints cannot easily be removed from the active set. In IPOPT (Wächter and Biegler, 2006), a large-scale interior point method, degenerate constraints are instead dealt with using regularization techniques (Wang et al., 2013; Chiang and Zavala, 2014). However, neither of these strategies are fully effective, as shown in the case study (Section 3).

With global degeneracies, the best option is model reformulation. For example, consider the system in Figure 1. Either the recycle pressure constraint, (3g), should be removed and all of the inequality constraints converted to equality constraints, or a pump should be added to the recycle loop. Identifying the cause of degenerate constraints such as these, however, is difficult in large problems with thousands of constraints.

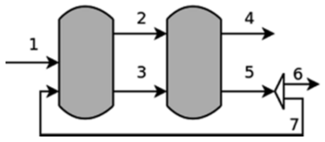

Figure 1: Pressure recycle degeneracy example

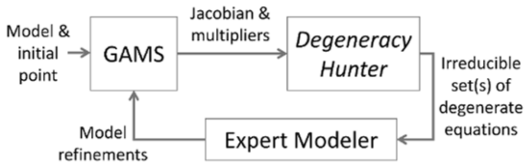

Figure 2: Recommend workflow with *Degeneracy Hunter*

Although factorization of the active Jacobian is sufficient for the identification of individual degenerate equations, it is still difficult to debug large models. Furthermore, blindly removing degenerate equation permanently from a model may be dangerous. For example, this could violate mass conservation when applied to (1). Instead *Degeneracy Hunter* aims to help expert modelers uncover the cause of degeneracies by finding the smallest sets of degenerate equations, i.e. **irreducible degenerate sets**. This allows the modeler to focus on a handful of equations instead of thousands (in large problems). Thus, the algorithms in *Degeneracy Hunter* are intended for post optimization analysis and not realtime use embedded in a NLP solver. The envisioned workflow with *Degeneracy Hunter* is shown in Figure 2.

2. The *Degeneracy Hunger* Algorithm

The *Degeneracy Hunter* algorithm is divided into four steps, and is summarized in Algorithm 1. First, in the **processing step**, the Jacobian and KKT (Karush-Kuhn-Tucker) multipliers are extracted from GAMS (or a similar environment), and the constraints are classified into four categories: equality, inactive inequality, strongly active inequality (non-zero multipliers) and weakly active inequality (zero multipliers). The *active Jacobian* (A_{dh}) is then assembled. It contains only the equality, strongly active inequality and optionally weakly active inequality constraints.

Next, in the **factorization step**, non-pivot rows of the active Jacobian are identified using either sparse or dense QR factorization. These rows are candidate degenerate equations. If the active Jacobian is not degenerate, there are no candidates and the algorithm terminates in this step.

In the **analysis step**, irreducible degenerate sets are calculated by solving a sequence of mixed integer linear programs (MILP), shown in (4), in order to minimize the number of non-zero elements in the adjoint vector λ. Integer variables y are used to count the number of non-zero elements. In order to avoid the trivial solution, $\|\lambda\| = 0$, the problem is resolved repeatedly with $\lambda_j = 1$ where $j = 1, \ldots n_{cand}$ are the indices of the n_{cand} candidate degenerate equations. Thus, this procedure may identify several irreducible degenerate sets instead of simply the smallest set. This is desirable, as there may be several independent sources of degeneracies in a large problem. Furthermore, because (4) is an MILP, its solution has the lowest possible cardinality. Through careful integration with mixed integer programming solvers, it would be possible to recover several irreducible degenerate sets containing candidate j with equal cardinality. Alternatively, it is also common for (4) to be infeasible. This indicates that candidate equation j does not significantly contribute to any degenerate set. Its original identification may have resulted from numerical noise in the factorization step. In the current implementation, the MILP is solved in GAMS with CPLEX or a similar solver. For the test problems, including the case study with 15,000+ variables, the MILPs solve in (typically far) less than 1 CPU minute each. This is due both to the efficiency of commercial MILP solvers and the sparsity of the Jacobian for many chemical engineering problems.

$$\min \quad \sum_{i=1}^{n_{rows}} y_i \tag{4a}$$

$$\text{s.t.} \quad A_{dh}^T \lambda = 0 \tag{4b}$$

$$-My_i \le \lambda_i \le My_i \ , \quad \forall i = 1, ..., n_{rows} \tag{4c}$$

$$\lambda_j = 1 \tag{4d}$$

Finally, in the **display step**, each irreducible degenerate set is reported (with the equation names from GAMS) along with the adjoint vector elements, λ, for these equations. This information allows the modeler to understand how a small number of equations interact to form degeneracies.

```
Data: Jacobian $A$, KKT multipliers $m$, constraint values $c$
Step 1: Assemble active Jacobian $A_{dh}$ using $A$, $m$, $c$ and user specified options ;
Step 2: Factorize $A_{dh}$ and identify set of candidate degenerate equations $\{S\}$ ;
foreach $j \in \{S\}$ do
    Step 3: Solve (4) ;
    if (4) is feasible then
        Step 4: Display non-zero elements of $\lambda$ (i.e. $y_i = 1$) and associated equation names ;
    else
        Step 4: Display equation $j$'s name and "is not part of a degenerate set"
    end
end
```

Algorithm 1: Pseudo-code for *Degeneracy Hunter*

3. Case Study: Air Separation Unit Design and Optimization

The authors previously developed an equation-based framework for flowsheet optimization in GAMS, and applied it to design air separation units for oxy-fired advanced power systems (Dowling and Biegler, 2015). The framework includes four key features: [1] embedded cubic equation of state thermodynamic calculations with vanishing and reappearing phases; [2] a novel equilibrium based distillation model with tray bypasses instead of integer variables; [3] simultaneous heat integration and process optimization; and [4] a trust region optimization algorithm to incorporate complex reactor models (e.g. computational fluid dynamics) without exact derivatives. Complementarity constraints are used throughout the framework to model switches, such as (dis)appearing phases and equipment being (in)active. The framework also includes a sophisticated initialization and multi-start procedure. In summary, a sequence of NLPs is solved in the framework. First the flowsheet is optimized with ideal thermodynamics and shortcut distillation models. These results are used to initialize more complex models, including cubic equation of state thermodynamics and the MESH with bypass distillation model. Applied to the ASU design problems, this ultimately yields a nonlinear program 15,000+ variables and constraints and $\approx$500 degrees of freedom.

Degeneracy Hunter has been applied to this model, and identified three sources of degenerate equations, which are a consequence of modeler (human) error.

3.1. Overspecifications

In the final stage of the optimization workflow for the equation-based framework, each half side of a heat exchange unit (condensers, reboilers, heat exchangers) is decomposed into a specified number of subunits. This is done to refine the constant heat capacity assumption used in the heat integration model; with subunits, the heat capacity is now represented by a piecewise linear function. The subunits are spaced with equal temperature changes. For example, if a heat exchanger changes the temperature of a stream by 25 K, and it is decomposed into 5 subunits in series, each subunit would have a temperature difference of 5 K.

Unfortunately in this subunit model, there were two instances of overspecification. For the temperature differences, the following equations were originally used:

$$T_u^{in} - T_u^{out} = \Delta T, \quad \forall u \in \{\text{Heat Exchange Units}\} \tag{5a}$$

$$T_s^{in} - T_s^{out} = \Delta T / n, \quad \forall s(u) \in \{\text{Heat Exchange Subunits}\}(u) \tag{5b}$$

where n is the number of subunits (specified constant). Clearly, (5a) can be identified as degenerate and removed. (5b) is sufficient on its own to enforce the equal temperature spacing.

The second source of degenerate equations (overspecifications) comes from inequality constraints. Each subunit is defined by either heating or cooling, and the sign of the heat duty for each unit is restricted accordingly, as shown in (6a) & (6b). Furthermore, to assist with optimization inlet and outlet temperatures are also constrained (e.g. temperature may not decrease in a heating unit), as shown in (6c) & (6d). These additional equations are also redundant. When $Q > 0$, the extra equations are inactive, and when $Q = 0$, the equations create a local degeneracy.

$$Q^i n_h \geq 0, \; Q_h^{out} = 0, \quad \forall h \in \{\text{Heating Subunits}\} \tag{6a}$$

$$Q^i n_c = 0, \; Q_c^{out} \geq 0, \quad \forall c \in \{\text{Cooling Subunits}\} \tag{6b}$$

$$T_h^{in} \leq T_h^{out}, \; \forall h \in \{\text{Heating Subunits}\} \tag{6c}$$

$$T_c^{in} \geq T_c^{out}, \; \forall c \in \{\text{Cooling Subunits}\} \tag{6d}$$

Finally, there is also a third source of degenerate equations with a pressure recycle between the distillation cascades and reboiler, similar to Figure 1.

All three of these sources of degeneracy were identified using *Degeneracy Hunter*. Although these modeling mistakes seem trivial, they easily go undetected within an equation-based framework that has several modules spread over more than ten thousand of lines of GAMS code. Without a systematical debugging tool, it is difficult to diagnose poor optimizer performances as either inadequate initialization or the presence of degenerate equations.

3.2. Computational Results

The ASU design optimization problem from our previous study was solved for the same 288 initial points, with several different configurations for the degenerate equations (Dowling and Biegler, 2015). For all of these computational studies, a desktop computer, running Ubuntu Linux and GAMS 24.3.1 with a quad-core 2.8 GHz Intel i7 processor, was used. In order to quantify variations in timings due to background jobs, GAMS overhead, disk access, etc., several of the trials were repeated. The timings are consistent within a few seconds. The average CPU time for all 288 instances of the optimization problem are reported in Table 1 for each trial. These CPU times include the entire initialization procedure described by Dowling and Biegler (2015).

The best performance was obtained with all three sources of degeneracy removed (753.9 s) and the worst performance occurred with all three sources of degeneracies remaining in the optimization formulation (896.4 s). Thus with this case study, removing the degenerate equations decreased CPU time by 16%. Furthermore, the presence of degenerate constraints also impacted the termination status with CONOPT. The last two columns of Table 1 show the number optimal and only feasible solution points, as classified by CONOPT, for each trial. “High quality” solutions

Table 1: Comparison of CPU times for the ASU design optimization problem with various degenerate constraints removed.

Trial	Pressure recycle removed?	(5a) removed?	(6c) & (6d) removed?	Average CPU time	"High Quality" Optimal	"High Quality" Only Feasible
1	No	No	No	896.4 s	210	14
2	No	Yes	Yes	863.6 s	204	17
3a	Yes	No	No	791.2 s	215	20
3b	Yes	No	No	788.7 s	215	20
4a	Yes	Yes	No	817.5 s	205	13
4b	Yes	Yes	No	818.4 s	205	13
5a	Yes	No	Yes	858.0 s	204	23
5b	Yes	No	Yes	859.8 s	204	23
6a	Yes	Yes	Yes	753.3 s	214	18
6b	Yes	Yes	Yes	754.1 s	214	18
6c	Yes	Yes	Yes	754.4 s	214	18

are defined as completely heat integrated with no complementarity violations. CONOPT terminated at locally optimal points 214 times in trial six (all degeneracies removed). In contrast, with some degeneracies present in trial five, CONOPT terminated at locally optimal only 204 times. This performance difference is expected, as the KKT multipliers are not unique in the presence of degeneracies. For the feasible only points, the active-set strategy may have not removed all of the degenerate constraints. Furthermore, in trial four (218), CONOPT terminated at 17 more infeasible and/or not high quality points than trial three (235). This speaks to the complexity of NLP solution techniques and the initialization procedure used in the framework. It is likely the presence of degeneracies (especially the pressure recycle loop) causes the NLP solver, CONOPT, to take different convergence paths early in the initialization procedure. This can result in very different solutions for each initial point considered.

4. Conclusions

In summary, we develop and describe the *Degeneracy Hunter*, an algorithm for determining irreducible degenerate sets of equations. Calculation of these sets is formulated as a mixed integer linear program. This contribution is especially important for debugging complex nonlinear programs with thousands of constraints. As demonstrated in a process design case study, the presence of a few degenerate equations may have a dramatic impact on solution time with active set methods, such as CONOPT. With interior point methods, such as IPOPT, degenerate equations can be even more serious, and could cause the solver to terminate at infeasible points. *Degeneracy Hunter* is a model debugging tool that helps expert modelers to identify degenerate constraints, reformulate their problems, and improve the performance of the large-scale optimization strategy. We recommend using *Degeneracy Hunter* to diagnose poor performance in NLP solvers, especially due to cycling and termination at non-optimal feasible points as a result of small step sizes.

References

Chiang, N., Zavala, V. M., 2014. An inertia-free filter line-search algorithm for large-scale nonlinear programming, preprint ANL/MCS-P5197-0914. URL `http://www.mcs.anl.gov/papers/P5197-0914.pdf`

Dowling, A. W., Biegler, L. T., 2015. A framework for efficient large scale equation-oriented flowsheet optimization. Computers & Chemical Engineering 72, 3 – 20.

Drud, A. S., 1994. CONOPT – A large-scale GRG code. ORSA Journal on Computing 6 (2), 207–216.

Wächter, A., Biegler, L. T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming 106 (1), 25–57.

Wang, K., Shao, Z., Lang, Y., Qian, J., Biegler, L. T., 2013. Barrier nlp methods with structured regularization for optimization of degenerate optimization problems. Computers & Chemical Engineering 57, 24–29.