Numerical Maths

isakhammer

A20

Contents

1 Lecture 1

1.1 Practical Information

- Brynjulf Owren, room 1350, Sentralbygg 2, brynjulf.owren@ntnu.no
- Alvar Lindell, room 1201, Sentralbygg 2, alvar.lindell@ntnu.no

There will be a total of 6 assignment where 4 should be approved. It should be delivered in blackboard as a jupyter notebook file including some control questions.

- **Project 1** It counts 10 procent on the final grade, relativaly small work, but somewhat large assignment. Every student submits her own separate .ipynb file. Discuss problem if you like, but make your own write-up. Likely to be a topic of algebra. Deadline. 10-15 September.
- **Project 2** Counts 20 procent on the final grade. Group project 1-3 students. Numerical ODE and may some optimization.

Lecture contents of the course

- Introduction 3.6%
- Numerical linear algebra 21.4%
- Numerical ODE 28.6%
- Nonlinear Systems and Numerical Optization 7.1%

May be jupyter programming on the exam.

1.2 M2 Basic Linear Algebra

1.2.1 Background summary

Vectors. Most of the time we think of vectors as n-plets of real numbers.

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Vecotrs are columns vectors if row vectors are needed use.

$$v^T = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_n \end{bmatrix}$$

Linear Transformations are given by $A : \mathbb{R}^n \to \mathbb{R}^m$. These are represented ass $m \times n$ matrices. $A = ((a_{ij}))$ such that $1 \leq i \leq m$ and $1 \leq j \leq n$. Notation $A \in \mathbb{R}^{m \times n}$

$$(Av)_i = \sum_{j=1}^{n} a_{ij} v_j, \quad i = 1, \dots, m.$$

If $A = ((a_{ij}))$, B $((b_{ij}))$ then $A + B = C$, $C = ((c_{ij}))$, $c_{ij} = a_{ij} + b_{ij}$.
Given to matrices, $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$

$$\mathbb{R}^n \to \mathbb{R}^k \to \mathbb{R}^m$$

$$\mathbb{R}^n \to \mathbb{R}^m$$

$$(A \cdot B)_{ij} = \sum_{r=1}^{k} a_{ir} b_{ri}$$

Fix a way to have notation on top of arrow and a better snippet for the summation. Might also train making quick vector notations.

1.2.2 Linear Independence

Let assume that we have $v_1, \ldots, v_k$ be vectors in $\mathbb{R}^n$ and let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be scalar if

$$\sum_{n=1}^{k} \alpha_i v_i = 0 \quad \text{then is} \quad \alpha_1 = \alpha_2 = \ldots = 0$$

Then $v_1, v_2, \ldots, v_k$ is linear independent.

1.2.3 Inverse of an matrix

If there is a matrix $B \in \mathbb{R}^{n\times n}$ such that

$$A \cdot B = B \cdot A = I$$

Then B is the inverse of A. B is denoted $B = A^{-1}$ Basis of $\mathbb{R}^n$. Any set of n linearly independent vectors in $\mathbb{R}^n$ is called a basis.

1.2.4 Permutation Matrix

Permuation Matrix. Let $I \in \mathbb{R}^{n\times n}$ be the identity matrix. I has columns $e_1, e_2, \ldots, e_n$ where e_i is the i-th canonical unit vector

$$\begin{bmatrix} 0 & 0 & \ldots 1 \ldots 0 \end{bmatrix} = e^T$$

Let $p = \begin{bmatrix} i_1, i_2, \ldots, i_n \end{bmatrix}^T$ Be a permutation of the set $\{1, \ldots, n\}$ then

$$P = \begin{bmatrix} e_1 & e_2 & e_2 \end{bmatrix}$$

The permutation matrix.

Implement example snippet

The inverse of a permutation matrix in $P^{-1} = P^T$ and $\left(P^{-1}\right)_{ij} = P_{ji}$.

1.2.5 Types of Matrices

- Symmetric: $A^T = A$
- Skew symmetric: $A^T = -A$
- Orthogonal. $A^T A = I$

2 Lecture 3 - August 25 - 2020

2.1 Continuation of previous lecture

Lets find a practical computation of $p^{(0)}, p^{(1)}, \ldots$. Always start with $p^{(0)} = r^{(0)} = b - Ax^{(0)}$. Suppose that $p^{(0)}, \ldots, p^{(k)}$ have been found. Set $p^{(k+1)} = r^{(k+1)} - b_k p^{(k)}$. Require that

$$0 = \left\langle p^{(k)}, p^{(k+1)} \right\rangle_A = \left\langle p^{(k)}, r^{(k+1)} \right\rangle - \beta_k \left\langle p^{(k)}, p^{(k)} \right\rangle$$

$$\text{so} \quad \beta_k = \frac{\left\langle p^{(k)}, r^{(k+1)} \right\rangle_A}{\left\langle p^{(k)}, p^{(k)} \right\rangle_A}$$

Note that $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$ and

$$\begin{aligned} b - Ax^{(k+1)} &= b - Ax^{(k)} - \alpha_k Ap^{(k)} \\ &\underbrace{r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}}_{\text{essential}} \end{aligned}$$

Let $V_k = span\left\{p^{(0)}, \ldots, p^{(k)}\right\}$ and since $r^{(0)} = p^{(s)}, \quad r^{(k+1)} = p^{(k+1)} - \alpha_k Ap^{(k)}$, it happens that $Ap^{(k)} \in V_{k+1}$, we have

$$V_k = span\left\{r^{(0)}, \ldots, r^{(k)}\right\}$$

We want to prove that $\left\langle p^{(k+1)}, p^{(j)} \right\rangle = 0$ for $j = 0, \ldots, k-1$

$$\left\langle r^{(k+1)} - \beta_k p^{(k)}, p^{(j)} \right\rangle_A = \left\langle r^{(k+1)}, p^{(j)} \right\rangle - \beta_k \left\langle p^{(k)}, p^{(j)} \right\rangle_A$$

We know that

$$Ap^{(j)} \in V_{j+1}, \quad Ap^{(j)} = \sum_{e=0}^{j+1} c_e p^{(e)}$$

$$\left\langle r^{(k+1)}, p^{(j)} \right\rangle_A = \sum_{e=0}^{j=1} \left\langle r^{(k+1)}, c_e p^{(e)} \right\rangle$$

Chosing the search directions like this is corresponding to the Conjugate gradient method.

2.2 Conjugate Gradient Method Algorithm

$$
\begin{aligned}
&x^{(0)} \quad \text{is given} \\
&\quad r^{(0)} = b - A \cdot x^{(0)} \\
&\quad p^{(s)} = r^{(s)} \\
&\text{For } k = 0, 1, 2, \ldots \\
&\qquad \begin{cases}
\alpha & = \frac{p^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}} \\
x^{(k+1)} & = x^{(k)} + \alpha p^{(k)} \\
r^{(k+1)} & = r^{(k)} - \alpha_k A p^{(k)} \\
\beta_k & = \frac{\left(Ap^{(k)}\right)^T r^{(k+1)}}{\left(Ap^{(k)}\right)^T p^{(k)}} \\
p^{(k+1)} & = r^{(k+1)} - \beta_k p^{(k)}
\end{cases} .
\end{aligned}
$$

2.3 Simplification

We want to simplify the expression for α_k and β_k

$$
\begin{aligned}
p^{(k+1)} &= r^{(k+1)} - \beta_k p^{(k)} \\
p^{(k)} &= r^{(k)} - \beta_{k-1} p^{(k-1)} \qquad \Longrightarrow \text{ multiply } \quad r^{(k)T} \\
r^{(k)T} p^{(k)} &= \|r^{(k)}\|_2^2 - \beta_{k-1} r^{(k)T} p^{(k-1)} \\
\text{So} \quad \alpha_k &= \frac{\|r^{(k)}\|_2^2}{p^{(k)} A p^{(k)}} \\
r^{(k+1)T} p^{(k+1)} &= \|r^{(k+1)}\|^2 - \beta_k r^{(k+1)T} p^{(k)} \\
r^{(k)T} p^{(k+1)} &= -\beta_k r^{(k)T} p^{(k)} = -\beta_k \|r^{(k)}\|^2 = \|r^{(k+1)}\|^2
\end{aligned}
$$

In the end is the results

$$
\beta_k = -\frac{\|r^{(k)}\|^2}{\|r^{(k+1)}\|^2} \quad \text{and} \quad \alpha_k = \frac{\|r^{(k)}\|_2^2}{p^{(k)} A p^{(k)}}
$$

2.4 Modified algorithm

$$p^{(0)} = r^{(0)}$$

$$r_l = \|r^{(0)}\|^2$$

For $\quad k = 0, 1, 2, \ldots$

$$\begin{cases} v & = Ap^{(k)} \\ t & = p^{(k)T}v \qquad \rightarrow \text{saxpy} \\ \alpha_k & = \frac{r_l}{t} \qquad \rightarrow \text{saxpy} \\ x^{(k+1)} & = x^{(k)} + \alpha_k p^{(k)} \qquad \rightarrow \text{inner product} \\ r_c & = \|r^{(k+1)}\|^2 \qquad \rightarrow \text{saxpy} \\ p^{(k+1)} & = r^{(k+1)} + \frac{rc}{r_l}p^{(k)} \qquad \rightarrow \text{saxpy} \\ r_l & \leftarrow r_c \end{cases}$$

Operations done in the numerical method

$$A \times \text{vector} \quad \left[B\left(h^2\right) \quad \text{for full matrices}\right]$$

2.5 Convergence of the Conjugate Gradient Algorithm

Since all search directions are mutually A- orthogonal. theu are linearly indepedent. After n iterations, they span all of $\mathbb{R}^n$.. Since the residuals $r^{(n)}$ is orthogonal to all of $r^{(0)}, \ldots, r^{(n-1)}$ must be 0 and therefore

$$0 = r^{(0)n} = b - Ax^{(n)} \rightarrow Ax^{(n)} = b$$

But then the algorithm is only competitive when it terminates in $k \ll n$ iterations with a sufficient accurate solution.

Theorem 2.1. *Let A be SPD. The error after k iterations is bounded as*

$$\|e^{(j)}\|_A \leq \frac{2c^k}{1+c^{2k}}, \quad c = \frac{\sqrt{K_2(A)}-1}{\sqrt{K_2(A)}+1}$$

Remark. $\|v\|_A = \sqrt{v^TAv}$

2.6 Next Lecture Hint

Next lecture will be about precondition. We solve $Ax = b$. An equivalent formulation is to pick an invertible P and solve

$$P^{-1}Ax = xP^{-1}$$

$$\hat{A}x = \hat{b}$$

Criteria

1. Let P approximate A
2. Should be cheap to solve systems

$$P^{-1}y = c$$

3 Lecture 01/09/20

Well posedness of the inital value problem.

$$\dot{y} = f(t, y), \quad y(0) = y_0$$

Is stable on $[0, T]$ if for any sufficiently small $\varepsilon > 0$ m there are $(\delta_0, \delta(t))$ s.t.

$$\|\delta_0\| < \varepsilon, \quad \|\delta(t)\| < \varepsilon, \quad t \in [0, T]$$

Such that $\|y(t) - z(t)\| < C \cdot \varepsilon \quad \forall t \in [0, T]$ for some constant C. $z(t)$ solves the IVP

$$\dot{z}(t) = f(t, z(t)) + \delta(t), \quad z_0 = y_0 + \delta_0$$

One can prove that if f is Lipschitz (constant L), then the solution is stable $C = (1+T)e^{tL}$.

3.1 Flow of a vector field

For us a vector field is a continuous map

$$f : \mathbb{R}^m \to \mathbb{R}^m$$

For a fixed value of t consider the map $\phi_{t,f}(y_0) = y(t)$ is called the t-flow of the vector field f. Its domain of definition may depend on t and f.

Suppose that

$$\phi_{t_1,f}(\phi_{t_2,f}(y_0)) = \phi_{t_1,t_2,f}(y_0)$$

and

$$\phi_{-t_1,f}(\phi_{t_1,f}(y_0)) = \phi_{0,f}(y_0) = y_0 \quad \Longrightarrow \quad (\phi_{t,f})^{-1} = \phi_{-t,f}$$

Typical notation:

$$\phi_{t,f}(y_0) = e^{tf} y_0$$

3.2 Numerical Integration of ODE

Always assume a finite time interval $[0, T]$.

Vector field (f) does not contain parameters. Split the finite interval into subintervals

$$t_0 < t_1 < \ldots < t^N = T$$

Often we assume $t_n = t_0 + nh$, where h is the stepsize. Also we may have $h = \frac{T - t_0}{N}$. For $t = t_j$, let $u_j \approx y(t_j) = y_j$ and $f_j f(t_j, y_j)$

Two classes of methods

1. One-step method , $u_{n+1} = \chi_{n+1}(u_n)$
2. Multistep methods, u_{n+1} depends on $u_n, u_{n-1}, \ldots, u_{nk+1}$ and also $f_{n+1}, f_n, \ldots, f_{n-k+1}$

3.2.1 The Simplest Schemes

1. **Euler** , $u_{n+1} = u_n + hf(t_n, u_n)$
2. **Backward Euler (Implicit Euler)**
$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$$
3. **Trapezoid rule** .
$$u_{n+1} = u_n + \frac{h}{2}(f())$$
4. Midpoint rule
$$u_{n+1} = u_n 0 hf\left(t_n + \frac{h}{2}, \frac{y_n + y_{n+1}}{2}\right)$$

3.2.2 The most important classes of one-step schemes

Taylor Series methods.

$$\dot{y} = f(t, y), \quad y(0) = y_0, \quad y(t) \in \mathbb{R}^m$$

If $y(t)$ is sufficiently smooth then we can compute the taylor expansion such that

$$y(t+h) = y(t)\, h\dot{y}(t) + \frac{1}{2}h^2\ddot{y}(t) + \ldots + \frac{1}{q!}h^q y^{(q)} + R_{q+1}.$$

where

$$R_{q+1} = \frac{y^{(q+1)}(\zeta)}{(q+1)!}, \quad \zeta \in (t, t+h)$$

Compute $y^{(k)}(t)$, $k > 1$ from diff eq.

$$\begin{aligned}\ddot{y}(t) &= \frac{d}{dt}f(t, y(t)) = \frac{\partial}{\partial t}f(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t))\,\dot{y}(t)\\ &= \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t))\end{aligned}$$

Similary we can compute

$$y^{(3)}(t), y^{(4)}(t), \ldots$$

and plug this into the taylor expansion and ignore the remainder term.

Example. Let $\dot{y} = y^2$ and $y(0) = y_0$ such that

$$\ddot{y} = 2y\dot{y} = 2y^3, \quad y^{(3)} = 6y^2\dot{y} = 6y^4, \quad y^{(k)} = k!y^{(k+1)}$$

This can be plugged into the taylor expansion such that

$$y(t+h) = y(t) + hy(t)^2 + \frac{1}{2}h^2 2y(t)^2 + \ldots = y(t) + hy(t)^2 + \ldots$$

which ends up with the method

$$y_{n+1} = \sum_{k=1}^{q} h^{k-1} y_n^k$$

3.2.3 Runga kutta methods

Generalization class of methods. COntains Euler, Backwards Euler, Trapozoidal Rule, Midpoint and more. The format is described like this

$$K_i = f\left(t_n + c_i h, u_n + \sum_{j=1}^{s} a_{ij} K_j\right) \quad i = 1, \dots, s$$

$$u_{n+1} = u_n + h \sum_{i=1}^{s} b_i K_i$$

- **Explicit RK Methods** . $a_{ij} = 0, \quad j \geq i$
- **Butcher Tablaeuz**

Table 1: Butcher Tablaeuz

c_1	a_{11}	$\dots$	a_{1s}
$\vdots$			
c_s	a_{s1}	$\dots$	a_{ss}
	b_1	$\dots$	b_s

3.3 Analysis of one-step methods

The exact solution does not obey the numerical formula. We write the formula as

$$u_{n+1} = u_n + h\phi_{h,f}(t_n, u_n)$$
$$y_{n+1} = y_n + h\phi_{h,f}(t_n, y_n) + \varepsilon_{n+1}$$

We set $\varepsilon_{n+1} = h\tau_{n+1}(h)$ where $\tau_{n+1}(h)$ is the local trunction error.

Define also

$$\tau(h) = \max_n \|\tau_{n+1}(h)\|$$

A method is called **consistent** if $\lim_{h\to 0} \tau(h) = 0$. If $\tau(h) = O(h^p)$ as $h \to 0$ then ot has order (of consistency) p.

4 References